NNAAMMEE

rsync - faster, flexible replacement for rcp

SSYYNNOOPPSSIISS

rsync [OPTION]... SRC [SRC]... DEST

rsync [OPTION]... SRC [SRC]... [USER@]HOST:DEST

rsync [OPTION]... SRC [SRC]... [USER@]HOST::DEST

rsync [OPTION]... SRC [SRC]... rsync://[USER@]HOST[:PORT]/DEST

rsync [OPTION]... SRC

rsync [OPTION]... [USER@]HOST:SRC [DEST]

rsync [OPTION]... [USER@]HOST::SRC [DEST]

rsync [OPTION]... rsync://[USER@]HOST[:PORT]/SRC [DEST]

DDEESSCCRRIIPPTTIIOONN

rsync is a program that behaves in much the same way that rcp does, but has many more options and uses the rsync remote-update protocol to greatly speed up file transfers when the destination file is being updated.

The rsync remote-update protocol allows rsync to transfer just the differences between two sets of files across the network connection, using an efficient checksum-search algorithm described in the technical report that accompanies this package.

Some of the additional features of rsync are:

- support for copying links, devices, owners, groups, and permissions

- exclude and exclude-from options similar to GNU tar
- a CVS exclude mode for ignoring the same files that CVS would ignore
- can use any transparent remote shell, including ssh or rsh
- does not require super-user privileges
- pipelining of file transfers to minimize latency costs
- support for anonymous or authenticated rsync daemons (ideal for mirroring)

GGEENNEERRAALL

Rsync copies files either to or from a remote host, or locally on the current host (it does not support copying files between two remote hosts).

There are two different ways for rsync to contact a remote system: using a remote-shell program as the transport (such as ssh or rsh) or contacting an rsync daemon directly via TCP. The remote-shell transport is used whenever the source or destination path contains a single colon (:) separator after a host specification. Contacting an rsync daemon directly happens when the source or destination path contains a double colon (::) separator after a host specification, OR when an rsync:// URL is specified (see also the "USING RSYNC-DAEMON FEATURES VIA A REMOTE-SHELL CONNECTION" section for an exception to this latter rule).

As a special case, if a single source arg is specified without a destination, the files are listed in an output format similar to "ls -l".

As expected, if neither the source or destination path specify a

remote
host, the copy occurs locally (see also the ----lliisstt--oonnllyy option).

SSEETTUUPP
See the file README for installation instructions.

Once installed, you can use rsync to any machine that you can access
via a remote shell (as well as some that you can access using the rsync
daemon-mode protocol). For remote transfers, a modern rsync uses ssh
for its communications, but it may have been configured to use a dif-
ferent remote shell by default, such as rsh or remsh.

You can also specify any remote shell you like, either by using the --ee
command line option, or by setting the RSYNC_RSH environment variable.

Note that rsync must be installed on both the source and destination
machines.

UUSSAAGGEE
You use rsync in the same way you use rcp. You must specify a source
and a destination, one of which may be remote.

Perhaps the best way to explain the syntax is with some examples:

```
rsync -t *.c foo:src/
```

This would transfer all files matching the pattern *.c from the current
directory to the directory src on the machine foo. If any of the files
already exist on the remote system then the rsync remote-update proto-
col is used to update the file by sending only the differences. See the
tech report for details.

```
rsync -avz foo:src/bar /data/tmp
```

This would recursively transfer all files from the directory src/bar on
the machine foo into the /data/tmp/bar directory on the local machine.
The files are transferred in "archive" mode, which ensures that sym-
bolic links, devices, attributes, permissions, ownerships, etc. are
preserved in the transfer. Additionally, compression will be used to
reduce the size of data portions of the transfer.

```
rsync -avz foo:src/bar/ /data/tmp
```

A trailing slash on the source changes this behavior to avoid creating
an additional directory level at the destination. You can think of a
trailing / on a source as meaning "copy the contents of this directory"
as opposed to "copy the directory by name", but in both cases the
attributes of the containing directory are transferred to the contain-
ing directory on the destination. In other words, each of the follow-
ing commands copies the files in the same way, including their setting
of the attributes of /dest/foo:

```
rsync -av /src/foo /dest
rsync -av /src/foo/ /dest/foo
```

Note also that host and module references don't require a trailing
slash to copy the contents of the default directory. For example, both
of these copy the remote directory's contents into "/dest":

```
rsync -av host: /dest
rsync -av host::module /dest
```

You can also use rsync in local-only mode, where both the source and
destination don't have a ':' in the name. In this case it behaves like

an improved copy command.

Finally, you can list all the (listable) modules available from a par-
ticular rsync daemon by leaving off the module name:

```
rsync somehost.mydomain.com::
```

See the following section for more details.

AADDVVAANNCCEEDD UUSSAAGGEE

The syntax for requesting multiple files from a remote host involves
using quoted spaces in the SRC. Some examples:

```
rsync host::'modname/dir1/file1 modname/dir2/file2' /dest
```

This would copy file1 and file2 into /dest from an rsync daemon. Each
additional arg must include the same "modname/" prefix as the first
one, and must be preceded by a single space. All other spaces are
assumed to be a part of the filenames.

```
rsync -av host:'dir1/file1 dir2/file2' /dest
```

This would copy file1 and file2 into /dest using a remote shell. This
word-splitting is done by the remote shell, so if it doesn't work it
means that the remote shell isn't configured to split its args based on
whitespace (a very rare setting, but not unknown). If you need to
transfer a filename that contains whitespace, you'll need to either
escape the whitespace in a way that the remote shell will understand,
or use wildcards in place of the spaces. Two examples of this are:

```
rsync -av host:'file\ name\ with\ spaces' /dest
rsync -av host:file?name?with?spaces /dest
```

This latter example assumes that your shell passes through

unmatched
wildcards. If it complains about "no match", put the name in quotes.

CCOONNNNEECCTTIINNGG TTOO AANN RRSSYYNNCC DDAAEEMMOONN

It is also possible to use rsync without a remote shell as the transport. In this case you will directly connect to a remote rsync daemon, typically using TCP port 873. (This obviously requires the daemon to be running on the remote system, so refer to the STARTING AN RSYNC DAEMON TO ACCEPT CONNECTIONS section below for information on that.)

Using rsync in this way is the same as using it with a remote shell except that:

- you either use a double colon :: instead of a single colon to separate the hostname from the path, or you use an rsync:// URL.
- the first word of the "path" is actually a module name.
- the remote daemon may print a message of the day when you connect.
- if you specify no path name on the remote daemon then the list of accessible paths on the daemon will be shown.
- if you specify no local destination then a listing of the specified files on the remote daemon is provided.
- you must not specify the ----rrsshh (--ee) option.

An example that copies all the files in a remote module named "src":

```
rsync -av host::src /dest
```

Some modules on the remote daemon may require authentication. If so,

you will receive a password prompt when you connect. You can avoid the password prompt by setting the environment variable RSYNC_PASSWORD to the password you want to use or using the ----ppaasssswwoorrdd--ffiillee option. This may be useful when scripting rsync.

WARNING: On some systems environment variables are visible to all users. On those systems using ----ppaasssswwoorrdd--ffiillee is recommended.

You may establish the connection via a web proxy by setting the environment variable RSYNC_PROXY to a hostname:port pair pointing to your web proxy. Note that your web proxy's configuration must support proxy connections to port 873.

UUSSIINNGG RRSSYYNNCC--DDAAEEMMOONN FFEEAATTUURREESS VVIIAA RREEMMOOTTEE--SSHHEELLLL CCOONNNNEECCTTIIOONN

It is sometimes useful to use various features of an rsync daemon (such as named modules) without actually allowing any new socket connections into a system (other than what is already required to allow remote-shell access). Rsync supports connecting to a host using a remote shell and then spawning a single-use "daemon" server that expects to read its config file in the home dir of the remote user. This can be useful if you want to encrypt a daemon-style transfer's data, but since the daemon is started up fresh by the remote user, you may not be able to use features such as chroot or change the uid used by the daemon. (For another way to encrypt a daemon transfer, consider using ssh to tunnel a local port to a remote machine and configure a normal rsync daemon on that remote host to only allow connections from "localhost".)

From the user's perspective, a daemon transfer via a remote-shell

con-
nection uses nearly the same command-line syntax as a normal rsync-
dae-
mon transfer, with the only exception being that you must
explicitly
set the remote shell program on the command-line with the ----
rrsshh==CCOOMMMMAANNDD
option. (Setting the RSYNC_RSH in the environment will not turn
on
this functionality.) For example:

rsync -av --rsh=ssh host::module /dest

If you need to specify a different remote-shell user, keep in mind
that
the user@ prefix in front of the host is specifying the rsync-
user
value (for a module that requires user-based authentication).
This
means that you must give the '-l user' option to ssh when
specifying
the remote-shell, as in this example that uses the short version of
the
----rrsshh option:

rsync -av -e "ssh -l ssh-user" rsync-user@host::module /dest

The "ssh-user" will be used at the ssh level; the "rsync-user" will
be
used to log-in to the "module".

SSTTAARRTTIINNGG AANN RRSSYYNNCC DDAAEEMMOONN TTOO AACCCCEEPPTT
CCOONNNNEECCTTIIOONNSS
In order to connect to an rsync daemon, the remote system needs to
have
a daemon already running (or it needs to have configured something
like
inetd to spawn an rsync daemon for incoming connections on a
particular
port). For full information on how to start a daemon that will
han-
dling incoming socket connections, see the rrssyynnccdd..ccoonnff(5)
man page --
that is the config file for the daemon, and it contains the
full
details for how to run the daemon (including stand-alone and inetd
con-

figurations).

If you're using one of the remote-shell transports for the transfer,
there is no need to manually start an rsync daemon.

EEXXAAMMPPLLEESS

Here are some examples of how I use rsync.

To backup my wife's home directory, which consists of large MS Word
files and mail folders, I use a cron job that runs

```
rsync -Cavz . arvidsjaur:backup
```

each night over a PPP connection to a duplicate directory on my machine
"arvidsjaur".

To synchronize my samba source trees I use the following Makefile tar-
gets:

```
get:
        rsync -avuzb --exclude '*~' samba:samba/ .
put:
        rsync -Cavuzb . samba:samba/
sync: get put
```

this allows me to sync with a CVS directory at the other end of the
connection. I then do CVS operations on the remote machine, which saves
a lot of time as the remote CVS protocol isn't very efficient.

I mirror a directory between my "old" and "new" ftp sites with the com-
mand:

```
rsync -az -e ssh --delete ~ftp/pub/samba nimbus:"~ftp/pub/tridge"
```

This is launched from cron every few hours.

OOPPTTIIOONNSS SSUUMMMMAARRYY

Here is a short summary of the options available in rsync. Please refer

```
       to the detailed description below for a complete description.

        -v, --verbose               increase verbosity
        -q, --quiet                 suppress non-error messages
            --no-motd               suppress daemon-mode MOTD (see caveat)
        -c, --checksum              skip based on checksum, not mod-time &
size
        -a, --archive               archive mode; same as -rlptgoD (no -H)
            --no-OPTION             turn off an implied OPTION (e.g. --no-
D)
        -r, --recursive             recurse into directories
        -R, --relative              use relative path names
            --no-implied-dirs       don't send implied dirs with --relative
        -b, --backup                make backups (see --suffix & --backup-
dir)
            --backup-dir=DIR        make backups into hierarchy based in
DIR
            --suffix=SUFFIX         backup suffix (default ~ w/o --backup-
dir)
        -u, --update                skip files that are newer on the
receiver
            --inplace               update destination files in-place
            --append                append data onto shorter files
        -d, --dirs                  transfer directories without recursing
        -l, --links                 copy symlinks as symlinks
        -L, --copy-links            transform symlink into referent file/
dir
            --copy-unsafe-links     only "unsafe" symlinks are transformed
            --safe-links            ignore symlinks that point outside the
tree
        -k, --copy-dirlinks         transform symlink to dir into referent
dir
        -K, --keep-dirlinks         treat symlinked dir on receiver as dir
        -H, --hard-links            preserve hard links
        -p, --perms                 preserve permissions
            --executability         preserve executability
            --chmod=CHMOD           affect file and/or directory
permissions
        -o, --owner                 preserve owner (super-user only)
        -g, --group                 preserve group
            --devices               preserve device files (super-user only)
            --specials              preserve special files
        -D                          same as --devices --specials
        -t, --times                 preserve times
        -O, --omit-dir-times        omit directories when preserving times
            --super                 receiver attempts super-user activities
        -S, --sparse                handle sparse files efficiently
        -n, --dry-run               show what would have been transferred
        -W, --whole-file            copy files whole (without rsync
algorithm)
```

```
        -x, --one-file-system       don't cross filesystem boundaries
        -B, --block-size=SIZE       force a fixed checksum block-size
        -e, --rsh=COMMAND           specify the remote shell to use
            --rsync-path=PROGRAM    specify the rsync to run on remote
machine
            --existing              skip creating new files on receiver
            --ignore-existing       skip updating files that exist on
receiver
            --remove-source-files   sender removes synchronized files (non-
dir)
            --del                   an alias for --delete-during
            --delete                delete extraneous files from dest dirs
            --delete-before         receiver deletes before transfer
(default)
            --delete-during         receiver deletes during xfer, not
before
            --delete-after          receiver deletes after transfer, not
before
            --delete-excluded       also delete excluded files from dest
dirs
            --ignore-errors         delete even if there are I/O errors
            --force                 force deletion of dirs even if not
empty
            --max-delete=NUM        don't delete more than NUM files
            --max-size=SIZE         don't transfer any file larger than
SIZE
            --min-size=SIZE         don't transfer any file smaller than
SIZE
            --partial               keep partially transferred files
            --partial-dir=DIR       put a partially transferred file into
DIR
            --delay-updates         put all updated files into place at end
        -m, --prune-empty-dirs      prune empty directory chains from file-
list
            --numeric-ids           don't map uid/gid values by user/group
name
            --timeout=TIME          set I/O timeout in seconds
        -I, --ignore-times          don't skip files that match size and
time
            --size-only             skip files that match in size
            --modify-window=NUM     compare mod-times with reduced accuracy
        -T, --temp-dir=DIR          create temporary files in directory DIR
        -y, --fuzzy                 find similar file for basis if no dest
file
            --compare-dest=DIR      also compare received files relative to
DIR
            --copy-dest=DIR         ... and include copies of unchanged
files
            --link-dest=DIR         hardlink to files in DIR when unchanged
        -z, --compress              compress file data during the transfer
```

```
            --compress-level=NUM    explicitly set compression level
        -C, --cvs-exclude           auto-ignore files in the same way CVS
does
        -f, --filter=RULE           add a file-filtering RULE
        -F                          same as --filter='dir-merge /.rsync-
filter'
                                    repeated: --filter='- .rsync-filter'
            --exclude=PATTERN       exclude files matching PATTERN
            --exclude-from=FILE     read exclude patterns from FILE
            --include=PATTERN       don't exclude files matching PATTERN
            --include-from=FILE     read include patterns from FILE
            --files-from=FILE       read list of source-file names from
FILE
        -0, --from0                 all *from/filter files are delimited by
0s
            --address=ADDRESS       bind address for outgoing socket to
daemon
            --port=PORT             specify double-colon alternate port
number
            --sockopts=OPTIONS      specify custom TCP options
            --blocking-io           use blocking I/O for the remote shell
            --stats                 give some file-transfer stats
        -8, --8-bit-output          leave high-bit chars unescaped in
output
        -h, --human-readable        output numbers in a human-readable
format
            --progress              show progress during transfer
        -P                          same as --partial --progress
        -i, --itemize-changes       output a change-summary for all updates
            --out-format=FORMAT     output updates using the specified
FORMAT
            --log-file=FILE         log what we're doing to the specified
FILE
            --log-file-format=FMT   log updates using the specified FMT
            --password-file=FILE    read password from FILE
            --list-only             list the files instead of copying them
            --bwlimit=KBPS          limit I/O bandwidth; KBytes per second
            --write-batch=FILE      write a batched update to FILE
            --only-write-batch=FILE like --write-batch but w/o updating
dest
            --read-batch=FILE       read a batched update from FILE
            --protocol=NUM          force an older protocol version to be
used
            --checksum-seed=NUM     set block/file checksum seed (advanced)
        -4, --ipv4                  prefer IPv4
        -6, --ipv6                  prefer IPv6
        -E, --extended-attributes   copy extended attributes, resource
forks
            --cache                 disable fcntl(F_NOCACHE)
            --version               print version number
```

```
       (-h) --help                  show this help (see below for -h
comment)


       Rsync  can also be run as a daemon, in which case the following
options
       are accepted:

            --daemon                run as an rsync daemon
            --address=ADDRESS       bind to the specified address
            --bwlimit=KBPS          limit I/O bandwidth; KBytes per second
            --config=FILE           specify alternate rsyncd.conf file
            --no-detach             do not detach from the parent
            --port=PORT             listen on alternate port number
            --log-file=FILE         override the "log file" setting
            --log-file-format=FMT   override the "log format" setting
            --sockopts=OPTIONS      specify custom TCP options
        -v, --verbose               increase verbosity
        -4, --ipv4                  prefer IPv4
        -6, --ipv6                  prefer IPv6
        -h, --help                  show this help (if used after --daemon)



OOPPTTIIOONNSS
       rsync uses the GNU long options  package.  Many  of  the  command
line
       options  have  two  variants,  one short and one long.  These are
shown
       below, separated by commas. Some options only have a long variant.
The
       '='  for  options  that take a parameter is optional; whitespace can
be
       used instead.


       ----hheellpp Print a short help page  describing  the  options
available  in
              rsync  and exit.  For backward-compatibility with older
versions
              of rsync, the help will also be output if you use the --hh
option
              without any other args.


       ----vveerrssiioonn
              print the rsync version number and exit.


       --vv,, ----vveerrbboossee
```

This option increases the amount of information you are given during the transfer. By default, rsync works silently. A single --vv will give you information about what files are being transferred and a brief summary at the end. Two --vv flags will give you information on what files are being skipped and slightly more information at the end. More than two --vv flags should only be used if you are debugging rsync.

Note that the names of the transferred files that are output are done using a default ----oouutt--ffoorrmmaatt of "%n%L", which tells you just the name of the file and, if the item is a link, where it points. At the single --vv level of verbosity, this does not mention when a file gets its attributes changed. If you ask for an itemized list of changed attributes (either ----iitteemmiizzee--cchhaannggeess or adding "%i" to the ----oouutt--ffoorrmmaatt setting), the output (on the client) increases to mention all items that are changed in any way. See the ----oouutt--ffoorrmmaatt option for more details.

--qq,, ----qquuiieett
This option decreases the amount of information you are given during the transfer, notably suppressing information messages from the remote server. This flag is useful when invoking rsync from cron.

----nnoo--mmoottdd
This option affects the information that is output by the client at the start of a daemon transfer. This suppresses the message-of-the-day (MOTD) text, but it also affects the list of

modules
that the daemon sends in response to the "rsync host::" request
(due to a limitation in the rsync protocol), so omit this option
if you want to request the list of modules from the deamon.

--II,, ----iiggnnoorree--ttiimmeess
Normally rsync will skip any files that are already the same
size and have the same modification time-stamp. This option
turns off this "quick check" behavior, causing all files to be
updated.

----ssiizzee--oonnllyy
Normally rsync will not transfer any files that are already the
same size and have the same modification time-stamp. With the
----ssiizzee--oonnllyy option, files will not be transferred if they have
the same size, regardless of timestamp. This is useful when
starting to use rsync after using another mirroring system which
may not preserve timestamps exactly.

----mmooddiiffyy--wwiinnddooww
When comparing two timestamps, rsync treats the timestamps as
being equal if they differ by no more than the modify-window
value. This is normally 0 (for an exact match), but you may
find it useful to set this to a larger value in some situations.
In particular, when transferring to or from an MS Windows FAT
filesystem (which represents times with a 2-second resolution),
----mmooddiiffyy--wwiinnddooww==11 is useful (allowing times to differ by up to 1
second).

--cc,, ----cchheecckkssuumm
This forces the sender to checksum _e_v_e_r_y regular file using a 128-bit MD4 checksum. It does this during the initial file-system scan as it builds the list of all available files. The receiver then checksums its version of each file (if it exists and it has the same size as its sender-side counterpart) in order to decide which files need to be updated: files with either a changed size or a changed checksum are selected for transfer. Since this whole-file checksumming of all files on both sides of the connection occurs in addition to the automatic checksum verifications that occur during a file's transfer, this option can be quite slow.

Note that rsync always verifies that each _t_r_a_n_s_f_e_r_r_e_d file was correctly reconstructed on the receiving side by checking its whole-file checksum, but that automatic after-the-transfer veri-fication has nothing to do with this option's before-the-trans-fer "Does this file need to be updated?" check.

--aa,, ----aarrcchhiivvee
This is equivalent to --rrllppttggooDD. It is a quick way of saying you want recursion and want to preserve almost everything (with -H being a notable omission). The only exception to the above equivalence is when ----ffiilleess--ffrroomm is specified, in which case --rr is not implied.

Note that --aa ddooeess nnoott pprreesseerrvvee hhaarrddlliinnkkss, because finding multi-ply-linked files is expensive. You must separately specify --HH.

--no-OPTION
You may turn off one or more implied options by prefixing the option name with "no-". Not all options may be prefixed with a "no-": only options that are implied by other options (e.g. ----nnoo--DD, ----nnoo--ppeerrmmss) or have different defaults in various circumstances (e.g. ----nnoo--wwhhoollee--ffiillee, ----nnoo--bbllooccckkiinngg--iioo, ----nnoo--ddiirrss).
You may specify either the short or the long option name after the "no-" prefix (e.g. ----nnoo--RR is the same as ----nnoo--rreellaattiivvee).

For example: if you want to use --aa (----aarrcchhiivvee) but don't want --oo (----oowwnneerr), instead of converting --aa into --rrllppttggDD, you could specify --aa ----nnoo--oo (or --aa ----nnoo--oowwnneerr).

The order of the options is important: if you specify ----nnoo--rr --aa, the --rr option would end up being turned on, the opposite of --aa ----nnoo--rr. Note also that the side-effects of the ----ffiilleess--ffrroomm option are NOT positional, as it affects the default state of several options and slightly changes the meaning of --aa (see the ----ffiilleess--ffrroomm option for more details).

--rr,, ----rreeccuurrssiivvee
This tells rsync to copy directories recursively. See also ----ddiirrss (--dd).

--RR,, ----rreellaattiivvee
Use relative paths. This means that the full path names specified on the command line are sent to the server rather than just the last parts of the filenames. This is particularly useful when you want to send several different directories at the

same
time. For example, if you used this command:

rsync -av /foo/bar/baz.c remote:/tmp/

... this would create a file named baz.c in /tmp/ on the
remote
machine. If instead you used

rsync -avR /foo/bar/baz.c remote:/tmp/

then a file named /tmp/foo/bar/baz.c would be created on
the
remote machine -- the full path name is preserved. To limit
the
amount of path information that is sent, you have a
couple
options: (1) With a modern rsync on the sending side
(beginning
with 2.6.7), you can insert a dot and a slash into the
source
path, like this:

rsync -avR /foo/./bar/baz.c remote:/tmp/

That would create /tmp/bar/baz.c on the remote machine.
(Note
that the dot must be followed by a slash, so "/foo/." would
not
be abbreviated.) (2) For older rsync versions, you would
need
to use a chdir to limit the source path. For example,
when
pushing files:

(cd /foo; rsync -avR bar/baz.c remote:/tmp/)

(Note that the parens put the two commands into a sub-shell,
so
that the "cd" command doesn't remain in effect for future
com-
mands.) If you're pulling files, use this idiom (which
doesn't
work with an rsync daemon):

rsync -avR --rsync-path="cd /foo; rsync" \

remote:bar/baz.c /tmp/

----nnoo--iimmpplliieedd--ddiirrss
This option affects the default behavior of the ----rreellaattiivvee
option. When it is specified, the attributes of the implied
directories from the source names are not included in the trans-
fer. This means that the corresponding path elements on the
destination system are left unchanged if they exist, and any
missing implied directories are created with default attributes.
This even allows these implied path elements to have big differ-
ences, such as being a symlink to a directory on one side of the
transfer, and a real directory on the other side.

For instance, if a command-line arg or a files-from entry told
rsync to transfer the file "path/foo/file", the directories
"path" and "path/foo" are implied when ----rreellaattiivvee is used. If
"path/foo" is a symlink to "bar" on the destination system, the
receiving rsync would ordinarily delete "path/foo", recreate it
as a directory, and receive the file into the new directory.
With ----nnoo--iimmpplliieedd--ddiirrss, the receiving rsync updates
"path/foo/file" using the existing path elements, which means
that the file ends up being created in "path/bar". Another way
to accomplish this link preservation is to use the
----kkeeeepp--ddiirrlliinnkkss option (which will also affect symlinks to
directories in the rest of the transfer).

In a similar but opposite scenario, if the transfer of
"path/foo/file" is requested and "path/foo" is a symlink on

the
sending side, running without ----nnoo--iimmpplliieedd--ddiirrss would cause
rsync to transform "path/foo" on the receiving side into an
identical symlink, and then attempt to transfer "path/foo/file",
which might fail if the duplicated symlink did not point to a
directory on the receiving side. Another way to avoid this
sending of a symlink as an implied directory is to use
----ccooppyy--uunnssaaffee--lliinnkkss, or ----ccooppyy--ddiirrlliinnkkss (both of which also
affect symlinks in the rest of the transfer -- see their
descriptions for full details).

--bb,, ----bbaacckkuupp
With this option, preexisting destination files are renamed as
each file is transferred or deleted. You can control where the
backup file goes and what (if any) suffix gets appended using
the ----bbaacckkuupp--ddiirr and ----ssuuffffiixx options.

Note that if you don't specify ----bbaacckkuupp--ddiirr, (1) the
----oommiitt--ddiirr--ttiimmeess option will be implied, and (2) if ----ddeelleettee is
also in effect (without ----ddeelleettee--eexxccluuddeedd), rsync will add a
"protect" filter-rule for the backup suffix to the end of all
your existing excludes (e.g. -f "P *~"). This will prevent pre-
viously backed-up files from being deleted. Note that if you
are supplying your own filter rules, you may need to manually
insert your own exclude/protect rule somewhere higher up in the
list so that it has a high enough priority to be effective
(e.g., if your rules specify a trailing inclusion/exclusion of
'*', the auto-added rule would never be reached).

----bbaacckkuupp--ddiirr==DDIIRR
In combination with the ----bbaacckkuupp option, this tells rsync to store all backups in the specified directory on the receiving side. This can be used for incremental backups. You can additionally specify a backup suffix using the ----ssuuffffiixx option (otherwise the files backed up in the specified directory will keep their original filenames).

----ssuuffffiixx==SSUUFFFFIIXX
This option allows you to override the default backup suffix used with the ----bbaacckkuupp (--bb) option. The default suffix is a ~ if no ---bbaacckkuupp--ddiirr was specified, otherwise it is an empty string.

--uu,, ----uuppddaattee
This forces rsync to skip any files which exist on the destination and have a modified time that is newer than the source file. (If an existing destination file has a modify time equal to the source file's, it will be updated if the sizes are different.)

In the current implementation of ----uuppddaattee, a difference of file format between the sender and receiver is always considered to be important enough for an update, no matter what date is on the objects. In other words, if the source has a directory or a symlink where the destination has a file, the transfer would occur regardless of the timestamps. This might change in the future (feel free to comment on this on the mailing list if you

have an opinion).

----iinnppllaaccee
This causes rsync not to create a new copy of the file and then move it into place. Instead rsync will overwrite the existing file, meaning that the rsync algorithm can't accomplish the full amount of network reduction it might be able to otherwise (since it does not yet try to sort data matches). One exception to this is if you combine the option with ----bbaacckkuupp, since rsync is smart enough to use the backup file as the basis file for the transfer.

This option is useful for transfer of large files with block-based changes or appended data, and also on systems that are disk bound, not network bound.

The option implies ----ppaarrttiiaall (since an interrupted transfer does not delete the file), but conflicts with ----ppaarrttiiaall--ddiirr and ----ddeellaayy--uuppddaatteess. Prior to rsync 2.6.4 ----iinnppllaaccee was also incompatible with ----ccoommppaarree--ddeesstt and ----lliinnkk--ddeesstt.

WARNING: The file's data will be in an inconsistent state during the transfer (and possibly afterward if the transfer gets interrupted), so you should not use this option to update files that are in use. Also note that rsync will be unable to update a file in-place that is not writable by the receiving user.

----aappppeennddd
This causes rsync to update a file by appending data onto the end of the file, which presumes that the data that

already
exists on the receiving side is identical with the start of the
file on the sending side. If that is not true, the file will
fail the checksum test, and the resend will do a normal
----iinnppllaaccee update to correct the mismatched data. Only files on
the receiving side that are shorter than the corresponding file
on the sending side (as well as new files) are sent. Implies
----iinnppllaaccee, but does not conflict with ----ssppaarrssee (though the
----ssppaarrssee option will be auto-disabled if a resend of the
already-existing data is required).

--dd,, ----ddiirrss
Tell the sending side to include any directories that are
encountered. Unlike ----rreeccuurrssiivvee, a directory's contents are not
copied unless the directory name specified is "." or ends with a
trailing slash (e.g. ".", "dir/.", "dir/", etc.). Without this
option or the ----rreeccuurrssiivvee option, rsync will skip all directo-
ries it encounters (and output a message to that effect for each
one). If you specify both ----ddiirrss and ----rreeccuurrssiivvee, ----rreeccuurrssiivvee
takes precedence.

--ll,, ----lliinnkkss
When symlinks are encountered, recreate the symlink on the des-
tination.

--LL,, ----ccooppyy--lliinnkkss
When symlinks are encountered, the item that they point to (the
referent) is copied, rather than the symlink. In older versions
of rsync, this option also had the side-effect of telling

the
receiving side to follow symlinks, such as symlinks to
directo-
ries. In a modern rsync such as this one, you'll need to
spec-
ify ----kkeeeepp--ddiirrlliinnkkss (--KK) to get this extra
behavior. The only
exception is when sending files to an rsync that is too old
to
understand --KK -- in that case, the --LL option will still
have the
side-effect of --KK on that older receiving rsync.

----ccooppyy--uunnssaaffee--lliinnkkss
This tells rsync to copy the referent of symbolic links
that
point outside the copied tree. Absolute symlinks are
also
treated like ordinary files, and so are any symlinks in
the
source path itself when ----rreellaattiivvee is used. This
option has no
additional effect if ----ccooppyy--lliinnkkss was also
specified.

----ssaaffee--lliinnkkss
This tells rsync to ignore any symbolic links which point
out-
side the copied tree. All absolute symlinks are also
ignored.
Using this option in conjunction with ----rreellaattiivvee
may give unex-
pected results.

--KK,, ----ccooppyy--ddiirrlliinnkkss
This option causes the sending side to treat a symlink
to a
directory as though it were a real directory. This is useful
if
you don't want symlinks to non-directories to be affected,
as
they would be using ----ccooppyy--lliinnkkss.

Without this option, if the sending side has replaced a
direc-
tory with a symlink to a directory, the receiving side
will

delete anything that is in the way of the new symlink, including
a directory hierarchy (as long as ----ffoorrccee or ----ddeelleettee is in
effect).

See also ----kkeeeepp--ddiirrlliinnkkss for an analogous option for the receiving side.

--KK,, ----kkeeeepp--ddiirrlliinnkkss
This option causes the receiving side to treat a symlink to a
directory as though it were a real directory, but only if it
matches a real directory from the sender. Without this option,
the receiver's symlink would be deleted and replaced with a real
directory.

For example, suppose you transfer a directory "foo" that contains a file "file", but "foo" is a symlink to directory "bar"
on the receiver. Without ----kkeeeepp--ddiirrlliinnkkss, the receiver deletes
symlink "foo", recreates it as a directory, and receives the
file into the new directory. With ----kkeeeepp--ddiirrlliinnkkss, the receiver
keeps the symlink and "file" ends up in "bar".

See also ----ccooppyy--ddiirrlliinnkkss for an analogous option for the sending side.

--HH,, ----hhaarrdd--lliinnkkss
This tells rsync to look for hard-linked files in the transfer
and link together the corresponding files on the receiving side.
Without this option, hard-linked files in the transfer are
treated as though they were separate files.

Note that rsync can only detect hard links if both parts of the

link are in the list of files being sent.

--pp,, ----ppeerrmmss
This option causes the receiving rsync to set the destination permissions to be the same as the source permissions. (See also the ----cchhmmoodd option for a way to modify what rsync considers to be the source permissions.)

When this option is _o_f_f, permissions are set as follows:

- Existing files (including updated files) retain their existing permissions, though the ----eexxeeccuuttaabbiilliittyy option might change just the execute permission for the file.
- New files get their "normal" permission bits set to the source file's permissions masked with the receiving end's umask setting, and their special permission bits disabled except in the case where a new directory inherits a set-gid bit from its parent directory.

Thus, when ----ppeerrmmss and ----eexxeeccuuttaabbiilliittyy are both disabled, rsync's behavior is the same as that of other file-copy utilities, such as ccpp(1) and ttaarr(1).

In summary: to give destination files (both old and new) the source permissions, use ----ppeerrmmss. To give new files the destination-default permissions (while leaving existing files unchanged), make sure that the ----ppeerrmmss option is off and use ----cchhmmoodd==uuggoo==rrwwXX (which ensures that all non-masked bits get enabled). If you'd care to make this latter behavior easier to

type, you could define a popt alias for it, such as putting this
line in the file ~/.popt (this defines the --ss option, and
includes --no-g to use the default group of the destination
dir):

rsync alias -s --no-p --no-g --chmod=ugo=rwX

You could then use this new option in a command such as this
one:

rsync -asv src/ dest/

(Caveat: make sure that --aa does not follow --ss, or it will re-
enable the "--no-*" options.)

The preservation of the destination's setgid bit on newly-cre-
ated directories when ----ppeerrmmss is off was added in rsync 2.6.7.
Older rsync versions erroneously preserved the three special
permission bits for newly-created files when ----ppeerrmmss was off,
while overriding the destination's setgid bit setting on a
newly-created directory. (Keep in mind that it is the version
of the receiving rsync that affects this behavior.)

----eexxeeccuuttaabbiilliittyy
This option causes rsync to preserve the executability (or non-
executability) of regular files when ----ppeerrmmss is not enabled. A
regular file is considered to be executable if at least one 'x'
is turned on in its permissions. When an existing destination
file's executability differs from that of the corresponding
source file, rsync modifies the destination file's permissions

as follows:

o To make a file non-executable, rsync turns off all its 'x' permissions.

o To make a file executable, rsync turns on each 'x' permission that has a corresponding 'r' permission enabled.

If ----ppeerrmmss is enabled, this option is ignored.

----cchhmmoodd

This option tells rsync to apply one or more comma-separated "chmod" strings to the permission of the files in the transfer. The resulting value is treated as though it was the permissions that the sending side supplied for the file, which means that this option can seem to have no effect on existing files if ----ppeerrmmss is not enabled.

In addition to the normal parsing rules specified in the cchhmmoodd(1) manpage, you can specify an item that should only apply to a directory by prefixing it with a 'D', or specify an item that should only apply to a file by prefixing it with a 'F'. For example:

--chmod=Dg+s,ug+w,Fo-w,+X

It is also legal to specify multiple ----cchhmmoodd options, as each additional option is just appended to the list of changes to make.

See the ----ppeerrmmss and ----eexxeeccuuttaabbiilliittyy options for how the result-

ing permission value can be applied to the files in the transfer.

--oo,, ----oowwnneerr
This option causes rsync to set the owner of the destination file to be the same as the source file, but only if the receiving rsync is being run as the super-user (see also the ----ssuuppeerr option to force rsync to attempt super-user activities). Without this option, the owner is set to the invoking user on the receiving side.

The preservation of ownership will associate matching names by default, but may fall back to using the ID number in some circumstances (see also the ----nnuummeerriicc--iiddss option for a full discussion).

--gg,, ----ggrroouupp
This option causes rsync to set the group of the destination file to be the same as the source file. If the receiving program is not running as the super-user (or if ----nnoo--ssuuppeerr was specified), only groups that the invoking user on the receiving side is a member of will be preserved. Without this option, the group is set to the default group of the invoking user on the receiving side.

The preservation of group information will associate matching names by default, but may fall back to using the ID number in some circumstances (see also the ----nnuummeerriicc--iiddss option for a full discussion).

----ddeevviicceess
This option causes rsync to transfer character and block device files to the remote system to recreate these devices. This option has no effect if the receiving rsync is not run as the super-user and ----ssuuppeerr is not specified.

----ssppeecciiaallss
This option causes rsync to transfer special files such as named sockets and fifos.

--DD The --DD option is equivalent to ----ddeevviicceess ----ssppeecciiaallss.

--tt,, ----ttiimmeess
This tells rsync to transfer modification times along with the files and update them on the remote system. Note that if this option is not used, the optimization that excludes files that have not been modified cannot be effective; in other words, a missing --tt or --aa will cause the next transfer to behave as if it used --II, causing all files to be updated (though the rsync algorithm will make the update fairly efficient if the files haven't actually changed, you're much better off using --tt).

--OO,, ----oommiitt--ddiirr--ttiimmeess
This tells rsync to omit directories when it is preserving modification times (see ----ttiimmeess). If NFS is sharing the directories on the receiving side, it is a good idea to use --OO. This option is inferred if you use ----bbaacckkuupp without ----bbaacckkuupp--ddiirr.

----ssuuppeerr
This tells the receiving side to attempt super-user activities even if the receiving rsync wasn't run by the super-user. These activities include: preserving users via the ----oowwnneerr option, preserving all groups (not just the current user's groups) via the ----ggrroouuppss option, and copying devices via the ----ddeevviicceess option. This is useful for systems that allow such activities without being the super-user, and also for ensuring that you will get errors if the receiving side isn't being running as the super-user. To turn off super-user activities, the super-user can use ----nnoo--ssuuppeerr.

--SS,, ----ssppaarrssee
Try to handle sparse files efficiently so they take up less space on the destination. Conflicts with ----iinnppllaaccee because it's not possible to overwrite data in a sparse fashion.

NOTE: Don't use this option when the destination is a Solaris "tmpfs" filesystem. It doesn't seem to handle seeks over null regions correctly and ends up corrupting the files.

--nn,, ----ddrryy--rruunn
This tells rsync to not do any file transfers, instead it will just report the actions it would have taken.

--WW,, ----wwhhoollee--ffiillee
With this option the incremental rsync algorithm is not used and the whole file is sent as-is instead. The transfer may be faster if this option is used when the bandwidth between the source and destination machines is higher than the bandwidth

to
disk (especially when the "disk" is actually a networked
filesystem). This is the default when both the source and destination are specified as local paths.

--xx,, ----oonnee--ffiillee--ssyyssttemm
This tells rsync to avoid crossing a filesystem boundary when
recursing. This does not limit the user's ability to specify
items to copy from multiple filesystems, just rsync's recursion
through the hierarchy of each directory that the user specified,
and also the analogous recursion on the receiving side during
deletion. Also keep in mind that rsync treats a "bind" mount to
the same device as being on the same filesystem.

If this option is repeated, rsync omits all mount-point directo-
ries from the copy. Otherwise, it includes an empty directory
at each mount-point it encounters (using the attributes of the
mounted directory because those of the underlying mount-point
directory are inaccessible).

If rsync has been told to collapse symlinks (via ----ccooppyy--lliinnkkss or
----ccooppyy--uunnssaaffee--lliinnkkss), a symlink to a directory on another device
is treated like a mount-point. Symlinks to non-directories are
unaffected by this option.

----eexxiissttiinngg,, ----iiggnnoorree--nnoonn--eexxiissttiinngg
This tells rsync to skip creating files (including directories)
that do not exist yet on the destination. If this option is
combined with the ----iiggnnoorree--eexxiissttiinngg option, no files will be
updated (which can be useful if all you want to do is to

delete
extraneous files).

----iiggnnoorree--eexxiissttiinngg
This tells rsync to skip updating files that already exist on
the destination (this does _n_o_t ignore existing directores, or
nothing would get done). See also ----eexxiissttiinngg.

----rreemmoovvee--ssoouurrccee--ffiilleess
This tells rsync to remove from the sending side the files
(meaning non-directories) that are a part of the transfer and
have been successfully duplicated on the receiving side.

----ddeelleettee
This tells rsync to delete extraneous files from the receiving
side (ones that aren't on the sending side), but only for the
directories that are being synchronized. You must have asked
rsync to send the whole directory (e.g. "dir" or "dir/") without
using a wildcard for the directory's contents (e.g. "dir/*")
since the wildcard is expanded by the shell and rsync thus gets
a request to transfer individual files, not the files' parent
directory. Files that are excluded from transfer are also
excluded from being deleted unless you use the ----
ddeelleettee--eexxcclluuddeedd
option or mark the rules as only matching on the sending side
(see the include/exclude modifiers in the FILTER RULES section).

Prior to rsync 2.6.7, this option would have no effect unless
----rreeccuurrssiivvee was in effect. Beginning with 2.6.7, deletions will
also occur when ----ddiirrss (--dd) is in effect, but only for directo-

ries whose contents are being copied.

This option can be dangerous if used incorrectly! It is a very good idea to run first using the ----ddrryy--rruunn option (--nn) to see what files would be deleted to make sure important files aren't listed.

If the sending side detects any I/O errors, then the deletion of any files at the destination will be automatically disabled. This is to prevent temporary filesystem failures (such as NFS errors) on the sending side causing a massive deletion of files on the destination. You can override this with the ----iiggnnoorree--eerrrroorrss option.

The ----ddeelleettee option may be combined with one of the --delete-WHEN options without conflict, as well as ----ddeelleettee--eexxcclluuddeedd. However, if none of the --delete-WHEN options are specified, rsync will currently choose the ----ddeelleettee--bbeeffoorree algorithm. A future version may change this to choose the ----ddeelleettee--dduurriinngg algorithm. See also ----ddeelleettee--aaffttteerr.

----ddeelleettee--bbeeffoorree
Request that the file-deletions on the receiving side be done before the transfer starts. This is the default if ----ddeelleettee or ----ddeelleettee--eexxcclluuddeedd is specified without one of the --delete-WHEN options. See ----ddeelleettee (which is implied) for more details on file-deletion.

Deleting before the transfer is helpful if the filesystem is tight for space and removing extraneous files would help to

make
 the transfer possible. However, it does introduce a
delay
 before the start of the transfer, and this delay might cause
the
 transfer to timeout (if ----ttiimmeeoouutt was specified).

 ----ddeelleettee--dduurriinngg,, ----ddeell
 Request that the file-deletions on the receiving side be
done
 incrementally as the transfer happens. This is a faster
method
 than choosing the before- or after-transfer algorithm, but it
is
 only supported beginning with rsync version 2.6.4. See ----
ddeelleettee
 (which is implied) for more details on file-deletion.

 ----ddeelleettee--aafftteerr
 Request that the file-deletions on the receiving side be
done
 after the transfer has completed. This is useful if you
are
 sending new per-directory merge files as a part of the
transfer
 and you want their exclusions to take effect for the
delete
 phase of the current transfer. See ----ddeelleettee (which
is implied)
 for more details on file-deletion.

 ----ddeelleettee--eexxcclluuddeedd
 In addition to deleting the files on the receiving side that
are
 not on the sending side, this tells rsync to also delete
any
 files on the receiving side that are excluded (see ----
eexxcclluuddee).
 See the FILTER RULES section for a way to make individual
exclu-
 sions behave this way on the receiver, and for a way to
protect
 files from ----ddeelleettee--eexxcclluuddeedd. See ----
ddeelleettee (which is implied)
 for more details on file-deletion.

----iiggnnoorree--eerrrroorrss
Tells ----ddeelleettee to go ahead and delete files even when there are
I/O errors.

----ffoorrccee
This option tells rsync to delete a non-empty directory when it
is to be replaced by a non-directory. This is only relevant if
deletions are not active (see ----ddeelleettee for details).

Note for older rsync versions: ----ffoorrccee used to still be required
when using ----ddeelleettee--aaffttteerr, and it used to be non-functional
unless the ----rreeccuurrssiivvee option was also enabled.

----mmaaxx--ddeelleettee==NNUUMM
This tells rsync not to delete more than NUM files or directo-
ries (NUM must be non-zero). This is useful when mirroring very
large trees to prevent disasters.

----mmaaxx--ssiizzee==SSIIZZEE
This tells rsync to avoid transferring any file that is larger
than the specified SIZE. The SIZE value can be suffixed with a
string to indicate a size multiplier, and may be a fractional
value (e.g. "----mmaaxx--ssiizzee==11..55mm").

The suffixes are as follows: "K" (or "KiB") is a kibibyte
(1024), "M" (or "MiB") is a mebibyte (1024*1024), and "G" (or
"GiB") is a gibibyte (1024*1024*1024). If you want the multi-
plier to be 1000 instead of 1024, use "KB", "MB", or "GB".
(Note: lower-case is also accepted for all values.) Finally, if
the suffix ends in either "+1" or "-1", the value will be offset
by one byte in the indicated direction.

Examples: --max-size=1.5mb-1 is 1499999 bytes, and
--max-size=2g+1 is 2147483649 bytes.

----mmiinn--ssiizzee==SSIIZZEE
This tells rsync to avoid transferring any file that is smaller
than the specified SIZE, which can help in not transferring
small, junk files. See the ----mmaaxx--ssiizzee option for a description
of SIZE.

--BB,, ----bblloocckk--ssiizzee==BBLLOOCCKKSSIIZZEE
This forces the block size used in the rsync algorithm to a
fixed value. It is normally selected based on the size of each
file being updated. See the technical report for details.

--ee,, ----rrsshh==CCOOMMMMAANNDD
This option allows you to choose an alternative remote shell
program to use for communication between the local and remote
copies of rsync. Typically, rsync is configured to use ssh by
default, but you may prefer to use rsh on a local network.

If this option is used with [[uusseerr@@]] hhoosstt::::mmoodduullee//ppaatthh, then the
remote shell _C_O_M_M_A_N_D will be used to run an rsync daemon on the
remote host, and all data will be transmitted through that
remote shell connection, rather than through a direct socket
connection to a running rsync daemon on the remote host. See
the section "USING RSYNC-DAEMON FEATURES VIA A REMOTE-SHELL CON-
NECTION" above.

Command-line arguments are permitted in COMMAND provided that
COMMAND is presented to rsync as a single argument. You

must
use spaces (not tabs or other whitespace) to separate the com-
mand and args from each other, and you can use single- and/or
double-quotes to preserve spaces in an argument (but not back-
slashes). Note that doubling a single-quote inside a single-
quoted string gives you a single-quote; likewise for double-
quotes (though you need to pay attention to which quotes your
shell is parsing and which quotes rsync is parsing). Some exam-
ples:

```
    -e 'ssh -p 2234'
    -e 'ssh -o "ProxyCommand nohup ssh firewall nc -w1 %h %p"'
```

(Note that ssh users can alternately customize site-specific
connect options in their .ssh/config file.)

You can also choose the remote shell program using the RSYNC_RSH
environment variable, which accepts the same range of values as
--ee.

See also the ----bbllooccckkiinngg--iioo option which is affected by this
option.

----rrssyynncc--ppaatthh==PPRROOGGRRAAMM
Use this to specify what program is to be run on the remote
machine to start-up rsync. Often used when rsync is not in the
default remote-shell's path (e.g.
--rsync-path=/usr/local/bin/rsync). Note that PROGRAM is run
with the help of a shell, so it can be any program, script, or
command sequence you'd care to run, so long as it does not cor-

rupt the standard-in & standard-out that rsync is using to com-
municate.

One tricky example is to set a different default directory on
the remote machine for use with the ----rreellaattiivvee option. For
instance:

```
rsync -avR --rsync-path="cd /a/b && rsync" hst:c/d /e/
```

--CC,, ----ccvvss--eexxcclluuddee
This is a useful shorthand for excluding a broad range of files
that you often don't want to transfer between systems. It uses
the same algorithm that CVS uses to determine if a file should
be ignored.

The exclude list is initialized to:

```
RCS SCCS CVS CVS.adm RCSLOG cvslog.* tags TAGS
.make.state .nse_depinfo *~ #* .#* ,* _$* *$ *.old *.bak
*.BAK *.orig *.rej .del-* *.a *.olb *.o *.obj *.so *.exe
*.Z *.elc *.ln core .svn/
```

then files listed in a $HOME/.cvsignore are added to the list
and any files listed in the CVSIGNORE environment variable (all
cvsignore names are delimited by whitespace).

Finally, any file is ignored if it is in the same directory as a
.cvsignore file and matches one of the patterns listed therein.
Unlike rsync's filter/exclude files, these patterns are split on
whitespace. See the ccvvss(1) manual for more information.

If you're combining --CC with your own ----ffiilltteerr rules, you should

note that these CVS excludes are appended at the end of your own rules, regardless of where the --CC was placed on the command-line. This makes them a lower priority than any rules you specified explicitly. If you want to control where these CVS excludes get inserted into your filter rules, you should omit the --CC as a command-line option and use a combination of ----ffiill--tteerr==::CC and ----ffiilltteerr==--CC (either on your command-line or by putting the ":C" and "-C" rules into a filter file with your other rules). The first option turns on the per-directory scanning for the .cvsignore file. The second option does a one-time import of the CVS excludes mentioned above.

--ff,, ----ffiilltteerr==RRUULLEE
This option allows you to add rules to selectively exclude certain files from the list of files to be transferred. This is most useful in combination with a recursive transfer.

You may use as many ----ffiilltteerr options on the command line as you like to build up the list of files to exclude.

See the FILTER RULES section for detailed information on this option.

--FF The --FF option is a shorthand for adding two ----ffiilltteerr rules to your command. The first time it is used is a shorthand for this rule:

--filter='dir-merge /.rsync-filter'

This tells rsync to look for per-directory .rsync-filter files

that have been sprinkled through the hierarchy and use their
rules to filter the files in the transfer. If --FF is repeated,
it is a shorthand for this rule:

--filter='exclude .rsync-filter'

This filters out the .rsync-filter files themselves from the
transfer.

See the FILTER RULES section for detailed information on how
these options work.

----eexxcclluuddee==PPAATTTTEERRNN
This option is a simplified form of the ----ffiilltteerr option that
defaults to an exclude rule and does not allow the full rule-
parsing syntax of normal filter rules.

See the FILTER RULES section for detailed information on this
option.

----eexxcclluuddee--ffrroomm==FFIILLEE
This option is related to the ----eexxcclluuddee option, but it specifies
a FILE that contains exclude patterns (one per line). Blank
lines in the file and lines starting with ';' or '#' are
ignored. If _F_I_L_E is --, the list will be read from standard
input.

----iinncclluuddee==PPAATTTTEERRNN
This option is a simplified form of the ----ffiilltteerr option that
defaults to an include rule and does not allow the full rule-
parsing syntax of normal filter rules.

See the FILTER RULES section for detailed information on

this
option.

----iinncclluuddee--ffrroomm==FFIILLEE
This option is related to the ----iinncclluuddee option, but it specifies
a FILE that contains include patterns (one per line). Blank
lines in the file and lines starting with ';' or '#' are
ignored. If _F_I_L_E is --, the list will be read from standard
input.

----ffiilleess--ffrroomm==FFIILLEE
Using this option allows you to specify the exact list of files
to transfer (as read from the specified FILE or -- for standard
input). It also tweaks the default behavior of rsync to make
transferring just the specified files and directories easier:

- The ----rreellaattiivvee (--RR) option is implied, which preserves the path information that is specified for each item in the file (use ----nnoo--rreellaattiivvee or ----nnoo--RR if you want to turn that off).
- The ----ddiirrss (--dd) option is implied, which will create directories specified in the list on the destination rather than noisily skipping them (use ----nnoo--ddiirrss or ----nnoo--dd if you want to turn that off).
- The ----aarrcchhiivvee (--aa) option's behavior does not imply ----rreeccuurrssiivvee (--rr), so specify it explicitly, if you want it.
- These side-effects change the default state of rsync, so

the position of the ----ffiilleess--ffrroomm option on the command-
line has no bearing on how other options are parsed (e.g.
--aa works the same before or after ----ffiilleess--ffrroomm, as does
----nnoo--RR and all other options).

The file names that are read from the FILE are all relative to
the source dir -- any leading slashes are removed and no ".."
references are allowed to go higher than the source dir. For
example, take this command:

```
rsync -a --files-from=/tmp/foo /usr remote:/backup
```

If /tmp/foo contains the string "bin" (or even "/bin"), the
/usr/bin directory will be created as /backup/bin on the remote
host. If it contains "bin/" (note the trailing slash), the
immediate contents of the directory would also be sent (without
needing to be explicitly mentioned in the file -- this began in
version 2.6.4). In both cases, if the --rr option was enabled,
that dir's entire hierarchy would also be transferred (keep in
mind that --rr needs to be specified explicitly with ----ffiilleess--ffrroomm,
since it is not implied by --aa). Also note that the effect of
the (enabled by default) ----rreellaattiivvee option is to duplicate only
the path info that is read from the file -- it does not force
the duplication of the source-spec path (/usr in this case).

In addition, the ----ffiilleess--ffrroomm file can be read from the remote
host instead of the local host if you specify a "host:" in front
of the file (the host must match one end of the transfer). As a

short-cut, you can specify just a prefix of ":" to mean "use the remote end of the transfer". For example:

rsync -a --files-from=:/path/file-list src:/ /tmp/copy

This would copy all the files specified in the /path/file-list file that was located on the remote "src" host.

--00,, ----ffrroomm00
This tells rsync that the rules/filenames it reads from a file are terminated by a null ('\0') character, not a NL, CR, or CR+LF. This affects ----eexxcclluuddee--ffrroomm, ----iinncclluuddee--ffrroomm, ----ffiilleess--ffrroomm, and any merged files specified in a ----ffiilltteerr rule.
It does not affect ----ccvvss--eexxcclluuddee (since all names read from a .cvsignore file are split on whitespace).

--TT,, ----tteemmpp--ddiirr==DDIIRR
This option instructs rsync to use DIR as a scratch directory when creating temporary copies of the files transferred on the receiving side. The default behavior is to create each tempo-rary file in the same directory as the associated destination file.

This option is most often used when the receiving disk partition does not have enough free space to hold a copy of the largest file in the transfer. In this case (i.e. when the scratch directory in on a different disk partition), rsync will not be able to rename each received temporary file over the top of the associated destination file, but instead must copy it into place. Rsync does this by copying the file over the top of

the
destination file, which means that the destination file
will
contain truncated data during this copy. If this were not
done
this way (even if the destination file were first removed,
the
data locally copied to a temporary file in the
destination
directory, and then renamed into place) it would be possible
for
the old file to continue taking up disk space (if someone had
it
open), and thus there might not be enough room to fit the
new
version on the disk at the same time.

If you are using this option for reasons other than a
shortage
of disk space, you may wish to combine it with
the
----ddeellaayy--uuppddaatteess option, which will ensure
that all copied files
get put into subdirectories in the destination hierarchy,
await-
ing the end of the transfer. If you don't have enough room
to
duplicate all the arriving files on the destination
partition,
another way to tell rsync that you aren't overly concerned
about
disk space is to use the ----ppaarrttiiaall--ddiirr option
with a relative
path; because this tells rsync that it is OK to stash off a
copy
of a single file in a subdir in the destination hierarchy,
rsync
will use the partial-dir as a staging area to bring over
the
copied file, and then rename it into place from there.
(Specify-
ing a ----ppaarrttiiaall--ddiirr with an absolute path
does not have this
side-effect.)

--yy,, ----ffuuzzzzyy
This option tells rsync that it should look for a basis file
for
any destination file that is missing. The current

algorithm
looks in the same directory as the destination file for either a
file that has an identical size and modified-time, or a simi-
larly-named file. If found, rsync uses the fuzzy basis file to
try to speed up the transfer.

Note that the use of the ----ddeelleettee option might get rid of any
potential fuzzy-match files, so either use ----ddeelleettee--aaffttteerr or
specify some filename exclusions if you need to prevent this.

----ccoommppaarree--ddeesstt==DDIIRR
This option instructs rsync to use _D_I_R on the destination
machine as an additional hierarchy to compare destination files
against doing transfers (if the files are missing in the desti-
nation directory). If a file is found in _D_I_R that is identical
to the sender's file, the file will NOT be transferred to the
destination directory. This is useful for creating a sparse
backup of just files that have changed from an earlier backup.

Beginning in version 2.6.4, multiple ----ccoommppaarree--ddeesstt directories
may be provided, which will cause rsync to search the list in
the order specified for an exact match. If a match is found
that differs only in attributes, a local copy is made and the
attributes updated. If a match is not found, a basis file from
one of the _D_I_Rs will be selected to try to speed up the trans-
fer.

If _D_I_R is a relative path, it is relative to the destination
directory. See also ----ccooppyy--ddeesstt and ----lliinnkk--ddeesstt.

----ccooppyy--ddeesstt==DDIIRR
This option behaves like ----ccoommppaarree--ddeesstt, but rsync will also
copy unchanged files found in _D_I_R to the destination directory
using a local copy. This is useful for doing transfers to a new
destination while leaving existing files intact, and then doing
a flash-cutover when all files have been successfully trans-
ferred.

Multiple ----ccooppyy--ddeesstt directories may be provided, which will
cause rsync to search the list in the order specified for an
unchanged file. If a match is not found, a basis file from one
of the _D_I_Rs will be selected to try to speed up the transfer.

If _D_I_R is a relative path, it is relative to the destination
directory. See also ----ccoommppaarree--ddeesstt and ----lliinnkk--ddeesstt.

----lliinnkk--ddeesstt==DDIIRR
This option behaves like ----ccooppyy--ddeesstt, but unchanged files are
hard linked from _D_I_R to the destination directory. The files
must be identical in all preserved attributes (e.g. permissions,
possibly ownership) in order for the files to be linked
together. An example:

```
rsync -av --link-dest=$PWD/prior_dir host:src_dir/ new_dir/
```

Beginning in version 2.6.4, multiple ----lliinnkk--ddeesstt directories may
be provided, which will cause rsync to search the list in the
order specified for an exact match. If a match is found that

differs only in attributes, a local copy is made and the
attributes updated. If a match is not found, a basis file from
one of the _D_I_Rs will be selected to try to speed up the trans-
fer.

Note that if you combine this option with ----iiggnnoorree--ttiimmeess, rsync
will not link any files together because it only links identical
files together as a substitute for transferring the file, never
as an additional check after the file is updated.

If _D_I_R is a relative path, it is relative to the destination
directory. See also ----ccoommppaarree--ddeesstt and ----ccooppyy--ddeesstt.

Note that rsync versions prior to 2.6.1 had a bug that could
prevent ----lliinnkk--ddeesstt from working properly for a non-super-user
when --oo was specified (or implied by --aa). You can work-around
this bug by avoiding the --oo option when sending to an old rsync.

--zz,, ----ccoommpprreessss
With this option, rsync compresses the file data as it is sent
to the destination machine, which reduces the amount of data
being transmitted -- something that is useful over a slow con-
nection.

Note that this option typically achieves better compression
ratios than can be achieved by using a compressing remote shell
or a compressing transport because it takes advantage of the
implicit information in the matching data blocks that are not
explicitly sent over the connection.

----ccoommpprreessss--lleevveell==NNUUMM
Explicitly set the compression level to use (see ----ccoommpprreessss)
instead of letting it default. If NUM is non-zero, the ----ccoomm--
pprreessss option is implied.

----nnuummeerriicc--iiddss
With this option rsync will transfer numeric group and user IDs
rather than using user and group names and mapping them at both
ends.

By default rsync will use the username and groupname to deter-
mine what ownership to give files. The special uid 0 and the
special group 0 are never mapped via user/group names even if
the ----nnuummeerriicc--iiddss option is not specified.

If a user or group has no name on the source system or it has no
match on the destination system, then the numeric ID from the
source system is used instead. See also the comments on the
"use chroot" setting in the rsyncd.conf manpage for information
on how the chroot setting affects rsync's ability to look up the
names of the users and groups and what you can do about it.

----ttiimmeeoouutt==TTIIMMEEOOUUTT
This option allows you to set a maximum I/O timeout in seconds.
If no data is transferred for the specified time then rsync will
exit. The default is 0, which means no timeout.

----aaddddrreessss
By default rsync will bind to the wildcard address when connect-
ing to an rsync daemon. The ----aaddddrreessss option allows you to

specify a specific IP address (or hostname) to bind to. See
also this option in the ----ddaaeemmoonn mode section.

----ppoorrtt==PPOORRTT
This specifies an alternate TCP port number to use rather than
the default of 873. This is only needed if you are using the
double-colon (::) syntax to connect with an rsync daemon (since
the URL syntax has a way to specify the port as a part of the
URL). See also this option in the ----ddaaeemmoonn mode section.

----ssoocckkooppttss
This option can provide endless fun for people who like to tune
their systems to the utmost degree. You can set all sorts of
socket options which may make transfers faster (or slower!).
Read the man page for the setsockopt() system call for details
on some of the options you may be able to set. By default no
special socket options are set. This only affects direct socket
connections to a remote rsync daemon. This option also exists
in the ----ddaaeemmoonn mode section.

----bbllooccckkiinngg--iioo
This tells rsync to use blocking I/O when launching a remote
shell transport. If the remote shell is either rsh or remsh,
rsync defaults to using blocking I/O, otherwise it defaults to
using non-blocking I/O. (Note that ssh prefers non-blocking
I/O.)

--ii,, ----iitteemmiizzee--cchhaannggeess
Requests a simple itemized list of the changes that are

being
made to each file, including attribute changes. This is exactly
the same as specifying ----oouutt--ffoorrmmaatt==''%%ii %%nn%%LL''. If you repeat
the option, unchanged files will also be output, but only if the
receiving rsync is at least version 2.6.7 (you can use --vvvv with
older versions of rsync, but that also turns on the output of
other verbose messages).

The "%i" escape has a cryptic output that is 9 letters long.
The general format is like the string YYXXccssttppooggzz, where YY is
replaced by the type of update being done, XX is replaced by the
file-type, and the other letters represent attributes that may
be output if they are being modified.

The update types that replace the YY are as follows:

- o A << means that a file is being transferred to the remote host (sent).
- o A >> means that a file is being transferred to the local host (received).
- o A cc means that a local change/creation is occurring for the item (such as the creation of a directory or the changing of a symlink, etc.).
- o A hh means that the item is a hard link to another item (requires ----hhaarrdd--lliinnkkss).
- o A .. means that the item is not being updated (though it might have attributes that are being modified).

The file-types that replace the XX are: ff for a file, a dd

for a
directory, an LL for a symlink, a DD for a device, and a SS for a
special file (e.g. named sockets and fifos).

The other letters in the string above are the actual letters
that will be output if the associated attribute for the item is
being updated or a "." for no change. Three exceptions to this
are: (1) a newly created item replaces each letter with a "+",
(2) an identical item replaces the dots with spaces, and (3) an
unknown attribute replaces each letter with a "?" (this can hap-
pen when talking to an older rsync).

The attribute that is associated with each letter is as follows:

- A cc means the checksum of the file is different and will be updated by the file transfer (requires ----cchheecckkssuumm).
- A ss means the size of the file is different and will be updated by the file transfer.
- A tt means the modification time is different and is being updated to the sender's value (requires ----ttiimmeess). An alternate value of TT means that the time will be set to the transfer time, which happens anytime a symlink is transferred, or when a file or device is transferred without ----ttiimmeess.
- A pp means the permissions are different and are being updated to the sender's value (requires ----ppeerrmmss).
- An oo means the owner is different and is being

updated to
the sender's value (requires ----oowwnneerr and super-user priv-
ileges).

- o A gg means the group is different and is being updated to the sender's value (requires ----ggrroouupp and the authority to set the group).
- o The zz slot is reserved for future use.

One other output is possible: when deleting files, the "%i" will output the string "*deleting" for each item that is being removed (assuming that you are talking to a recent enough rsync that it logs deletions instead of outputting them as a verbose message).

----oouutt--ffoorrmmaatt==FFOORRMMAATT

This allows you to specify exactly what the rsync client outputs to the user on a per-update basis. The format is a text string containing embedded single-character escape sequences prefixed with a percent (%) character. For a list of the possible escape characters, see the "log format" setting in the rsyncd.conf man-page.

Specifying this option will mention each file, dir, etc. that gets updated in a significant way (a transferred file, a recre-ated symlink/device, or a touched directory). In addition, if the itemize-changes escape (%i) is included in the string, the logging of names increases to mention any item that is changed in any way (as long as the receiving side is at least 2.6.4).

See the ----iitteemmiizzee--cchhaannggeess option for a description of the output of "%i".

The ----vveerrbboossee option implies a format of "%n%L", but you can use ----oouutt--ffoorrmmaatt without ----vveerrbboossee if you like, or you can override the format of its per-file output using this option.

Rsync will output the out-format string prior to a file's transfer unless one of the transfer-statistic escapes is requested, in which case the logging is done at the end of the file's transfer. When this late logging is in effect and ----pprrooggrreessss is also specified, rsync will also output the name of the file being transferred prior to its progress information (followed, of course, by the out-format output).

----lloogg--ffiillee==FFIILLEE

This option causes rsync to log what it is doing to a file. This is similar to the logging that a daemon does, but can be requested for the client side and/or the server side of a non-daemon transfer. If specified as a client option, transfer logging will be enabled with a default format of "%i %n%L". See the ----lloogg--ffiillee--ffoorrmmaatt option if you wish to override this.

Here's a example command that requests the remote side to log what is happening:

```
rsync -av --rsync-path="rsync --log-file=/tmp/rlog" src/ dest/
```

This is very useful if you need to debug why a connection is closing unexpectedly.

----lloogg--ffiillee--ffoorrmmaatt==FFOORRMMAATT
This allows you to specify exactly what per-update logging is put into the file specified by the ----lloogg--ffiillee option (which must also be specified for this option to have any effect). If you specify an empty string, updated files will not be mentioned in the log file. For a list of the possible escape characters, see the "log format" setting in the rsyncd.conf manpage.

----ssttaattss
This tells rsync to print a verbose set of statistics on the file transfer, allowing you to tell how effective the rsync algorithm is for your data.

The current statistics are as follows:

- NNuummbbeerr ooff ffiilleess is the count of all "files" (in the generic sense), which includes directories, symlinks, etc.
- NNuummbbeerr ooff ffiilleess ttrraannssffeerrrreedd is the count of normal files that were updated via the rsync algorithm, which does not include created dirs, symlinks, etc.
- TToottaall ffiillee ssiizzee is the total sum of all file sizes in the transfer. This does not count any size for directories or special files, but does include the size of symlinks.
- TToottaall ttrraannssffeerrrreedd ffiillee ssiizzee is the total sum of all files sizes for just the transferred files.
- LLiitteerraall ddaattaa is how much unmatched file-update data we

had to send to the receiver for it to recreate the
updated files.

o MMaattcchheedd ddaattaa is how much data the receiver got locally
when recreating the updated files.

o FFiillee lliisstt ssiizzee is how big the file-list data was when the
sender sent it to the receiver. This is smaller than the
in-memory size for the file list due to some compressing
of duplicated data when rsync sends the list.

o FFiillee lliisstt ggeenneerraattiioonn ttiimmee is the number of seconds that
the sender spent creating the file list. This requires a
modern rsync on the sending side for this to be present.

o FFiillee lliisstt ttrraannssffeerr ttiimmee is the number of seconds that the
sender spent sending the file list to the receiver.

o TTootttaall bbyytteess sseenntt is the count of all the bytes that rsync
sent from the client side to the server side.

o TTootttaall bbyytteess rreecceeiivveedd is the count of all non-message
bytes that rsync received by the client side from the
server side. "Non-message" bytes means that we don't
count the bytes for a verbose message that the server
sent to us, which makes the stats more consistent.

--88,, ----88--bbiitt--oouuttppuutt
This tells rsync to leave all high-bit characters unescaped in
the output instead of trying to test them to see if they're
valid in the current locale and escaping the invalid ones. All

control characters (but never tabs) are always escaped, regardless of this option's setting.

The escape idiom that started in 2.6.7 is to output a literal backslash (\) and a hash (#), followed by exactly 3 octal digits. For example, a newline would output as "\#012". A literal backslash that is in a filename is not escaped unless it is followed by a hash and 3 digits (0-9).

--hh,, ----hhuummaann--rreeaaddaabbllee
Output numbers in a more human-readable format. This makes big numbers output using larger units, with a K, M, or G suffix. If this option was specified once, these units are K (1000), M (1000*1000), and G (1000*1000*1000); if the option is repeated, the units are powers of 1024 instead of 1000.

----ppaarrttiiaall
By default, rsync will delete any partially transferred file if the transfer is interrupted. In some circumstances it is more desirable to keep partially transferred files. Using the ----ppaarr--ttiiaall option tells rsync to keep the partial file which should make a subsequent transfer of the rest of the file much faster.

----ppaarrttiiaall--ddiirr==DDIIRR
A better way to keep partial files than the ----ppaarrttiiaall option is to specify a _D_I_R that will be used to hold the partial data (instead of writing it out to the destination file). On the next transfer, rsync will use a file found in this dir as data to speed up the resumption of the transfer and then delete

it
after it has served its purpose.

Note that if ----wwhhoollee--ffiillee is specified (or implied), any par-
tial-dir file that is found for a file that is being updated
will simply be removed (since rsync is sending files without
using the incremental rsync algorithm).

Rsync will create the _D_I_R if it is missing (just the last dir --
not the whole path). This makes it easy to use a relative path
(such as "----ppaarrttiiaall--ddiirr==..rrssyynncc--ppaarrttiiaall") to have rsync create
the partial-directory in the destination file's directory when
needed, and then remove it again when the partial file is
deleted.

If the partial-dir value is not an absolute path, rsync will add
an exclude rule at the end of all your existing excludes. This
will prevent the sending of any partial-dir files that may exist
on the sending side, and will also prevent the untimely deletion
of partial-dir items on the receiving side. An example: the
above ----ppaarrttiiaall--ddiirr option would add the equivalent of
"----eexxcclluuddee==..rrssyynncc--ppaarrttiiaall//" at the end of any other filter
rules.

If you are supplying your own exclude rules, you may need to add
your own exclude/hide/protect rule for the partial-dir because
(1) the auto-added rule may be ineffective at the end of your
other rules, or (2) you may wish to override rsync's exclude
choice. For instance, if you want to make rsync clean-up any
left-over partial-dirs that may be lying around, you

should
specify ----ddeelleettee--aaffttteerr and add a "risk" filter rule, e.g. --ff ''RR
..rrssyynncc--ppaarrttiiaall//''. (Avoid using ----ddeelleettee--bbeeffoorree or ----ddeelleettee--dduurr--
iinngg unless you don't need rsync to use any of the left-over par-
tial-dir data during the current run.)

IMPORTANT: the ----ppaarrttiiaall--ddiirr should not be writable by other
users or it is a security risk. E.g. AVOID "/tmp".

You can also set the partial-dir value the RSYNC_PARTIAL_DIR
environment variable. Setting this in the environment does not
force ----ppaarrttiiaall to be enabled, but rather it affects where par-
tial files go when ----ppaarrttiiaall is specified. For instance,
instead of using ----ppaarrttiiaall--ddiirr==..rrssyynncc--ttmmpp along with ----pprrooggrreessss,
you could set RSYNC_PARTIAL_DIR=.rsync-tmp in your environment
and then just use the --PP option to turn on the use of the
.rsync-tmp dir for partial transfers. The only times that the
----ppaarrttiiaall option does not look for this environment value are
(1) when ----iinnppllaaccee was specified (since ----iinnppllaaccee conflicts with
----ppaarrttiiaall--ddiirr), and (2) when ----ddeellaayy--uuppddaatteess was specified (see
below).

For the purposes of the daemon-config's "refuse options" set-
ting, ----ppaarrttiiaall--ddiirr does _n_o_t imply ----ppaarrttiiaall. This is so that a
refusal of the ----ppaarrttiiaall option can be used to disallow the
overwriting of destination files with a partial transfer, while
still allowing the safer idiom provided by ----ppaarrttiiaall--ddiirr.

----ddeellaayy--uuppddaatteess

This option puts the temporary file from each updated file into a holding directory until the end of the transfer, at which time all the files are renamed into place in rapid succession. This attempts to make the updating of the files a little more atomic. By default the files are placed into a directory named ".~tmp~" in each file's destination directory, but if you've specified the ----ppaarrttiiaall--ddiirr option, that directory will be used instead. See the comments in the ----ppaarrttiiaall--ddiirr section for a discussion of how this ".~tmp~" dir will be excluded from the transfer, and what you can do if you wnat rsync to cleanup old ".~tmp~" dirs that might be lying around. Conflicts with ----iinnppllaaccee and ----aappppeennddd.

This option uses more memory on the receiving side (one bit per file transferred) and also requires enough free disk space on the receiving side to hold an additional copy of all the updated files. Note also that you should not use an absolute path to ----ppaarrttiiaall--ddiirr unless (1) there is no chance of any of the files in the transfer having the same name (since all the updated files will be put into a single directory if the path is abso-lute) and (2) there are no mount points in the hierarchy (since the delayed updates will fail if they can't be renamed into place).

See also the "atomic-rsync" perl script in the "support" subdir for an update algorithm that is even more atomic (it uses ----lliinnkk--ddeesstt and a parallel hierarchy of files).

--mm,, ----pprruunnee--eemmppttyy--ddiirrss
This option tells the receiving rsync to get rid of empty direc-
tories from the file-list, including nested directories that
have no non-directory children. This is useful for avoiding the
creation of a bunch of useless directories when the sending
rsync is recursively scanning a hierarchy of files using
include/exclude/filter rules.

Because the file-list is actually being pruned, this option also
affects what directories get deleted when a delete is active.
However, keep in mind that excluded files and directories can
prevent existing items from being deleted (because an exclude
hides source files and protects destination files).

You can prevent the pruning of certain empty directories from
the file-list by using a global "protect" filter. For instance,
this option would ensure that the directory "emptydir" was kept
in the file-list:

--filter 'protect emptydir/'

Here's an example that copies all .pdf files in a hierarchy,
only creating the necessary destination directories to hold the
.pdf files, and ensures that any superfluous files and directo-
ries in the destination are removed (note the hide filter of
non-directories being used instead of an exclude):

rsync -avm --del --include='*.pdf' -f 'hide,! */' src/ dest

If you didn't want to remove superfluous destination files, the

more time-honored options of "--include='*/' --exclude='*'" would work fine in place of the hide-filter (if that is more natural to you).

----pprrooggrreessss

This option tells rsync to print information showing the progress of the transfer. This gives a bored user something to watch. Implies ----vveerrbboossee if it wasn't already specified.

While rsync is transferring a regular file, it updates a progress line that looks like this:

```
      782448  63%  110.64kB/s    0:00:04
```

In this example, the receiver has reconstructed 782448 bytes or 63% of the sender's file, which is being reconstructed at a rate of 110.64 kilobytes per second, and the transfer will finish in 4 seconds if the current rate is maintained until the end.

These statistics can be misleading if the incremental transfer algorithm is in use. For example, if the sender's file consists of the basis file followed by additional data, the reported rate will probably drop dramatically when the receiver gets to the literal data, and the transfer will probably take much longer to finish than the receiver estimated as it was finishing the matched part of the file.

When the file transfer finishes, rsync replaces the progress line with a summary line that looks like this:

```
     1238099 100%  146.38kB/s    0:00:08  (xfer#5, to-check=169/396)
```

In this example, the file was 1238099 bytes long in total, the average rate of transfer for the whole file was 146.38 kilobytes per second over the 8 seconds that it took to complete, it was the 5th transfer of a regular file during the current rsync session, and there are 169 more files for the receiver to check (to see if they are up-to-date or not) remaining out of the 396 total files in the file-list.

--PP The --PP option is equivalent to ----ppaarrttiiaall ----pprrooggrreessss. Its purpose is to make it much easier to specify these two options for a long transfer that may be interrupted.

----ppaasssswwoorrdd--ffiillee
This option allows you to provide a password in a file for accessing a remote rsync daemon. Note that this option is only useful when accessing an rsync daemon using the built in transport, not when using a remote shell as the transport. The file must not be world readable. It should contain just the password as a single line.

----lliisstt--oonnllyy
This option will cause the source files to be listed instead of transferred. This option is inferred if there is a single source arg and no destination specified, so its main uses are: (1) to turn a copy command that includes a destination arg into a file-listing command, (2) to be able to specify more than one local source arg (note: be sure to include the destination),

or
(3) to avoid the automatically added "--rr ----eexxcclluuddee=='//**//**'" options that rsync usually uses as a compatibility kluge when generating a non-recursive listing. Caution: keep in mind that a source arg with a wild-card is expanded by the shell into multiple args, so it is never safe to try to list such an arg without using this option. For example:

rsync -av --list-only foo* dest/

----bbwwlliimmiitt==KKBBPPSS
This option allows you to specify a maximum transfer rate in kilobytes per second. This option is most effective when using rsync with large files (several megabytes and up). Due to the nature of rsync transfers, blocks of data are sent, then if rsync determines the transfer was too fast, it will wait before sending the next data block. The result is an average transfer rate equaling the specified limit. A value of zero specifies no limit.

----wwrriittee--bbaattcchh==FFIILLEE
Record a file that can later be applied to another identical destination with ----rreeaadd--bbaattcchh. See the "BATCH MODE" section for details, and also the ----oonnllyy--wwrriittee--bbaattcchh option.

----oonnllyy--wwrriittee--bbaattcchh==FFIILLEE
Works like ----wwrriittee--bbaattcchh, except that no updates are made on the destination system when creating the batch. This lets you transport the changes to the destination system via some

other
means and then apply the changes via ----rreeaadd--bbaattcchh.

Note that you can feel free to write the batch directly to some portable media: if this media fills to capacity before the end of the transfer, you can just apply that partial transfer to the destination and repeat the whole process to get the rest of the changes (as long as you don't mind a partially updated destination system while the multi-update cycle is happening).

Also note that you only save bandwidth when pushing changes to a remote system because this allows the batched data to be diverted from the sender into the batch file without having to flow over the wire to the receiver (when pulling, the sender is remote, and thus can't write the batch).

----rreeaadd--bbaattcchh==FFIILLEE
Apply all of the changes stored in FILE, a file previously generated by ----wwrriittee--bbaattcchh. If _F_I_L_E is --, the batch data will be read from standard input. See the "BATCH MODE" section for details.

----pprroottooccooll==NNUUMM
Force an older protocol version to be used. This is useful for creating a batch file that is compatible with an older version of rsync. For instance, if rsync 2.6.4 is being used with the ----wwrriittee--bbaattcchh option, but rsync 2.6.3 is what will be used to run the ----rreeaadd--bbaattcchh option, you should use "--protocol=28" when creating the batch file to force the older protocol version to

be used in the batch file (assuming you can't upgrade the rsync
on the reading system).

--44,, ----iippvv44 or --66,, ----iippvv66
Tells rsync to prefer IPv4/IPv6 when creating sockets. This
only affects sockets that rsync has direct control over, such as
the outgoing socket when directly contacting an rsync daemon.
See also these options in the ----ddaaeemmoonn mode section.

----cchheecckkssuumm--sseeeedd==NNUUMM
Set the MD4 checksum seed to the integer NUM. This 4 byte
checksum seed is included in each block and file MD4 checksum
calculation. By default the checksum seed is generated by the
server and defaults to the current time() . This option is used
to set a specific checksum seed, which is useful for applica-
tions that want repeatable block and file checksums, or in the
case where the user wants a more random checksum seed. Note
that setting NUM to 0 causes rsync to use the default of time()
for checksum seed.

--EE,, ----eexxtteennddeedd--aattttrriibbuutteess
Apple specific option to copy extended attributes, resource
forks, and ACLs. Requires at least Mac OS X 10.4 or suitably
patched rsync.

----ccaacchhee
Apple specific option to enable filesystem caching of rsync file
i/o Otherwise fcntl(F_NOCACHE) is used to limit memory growth.

DDAAEEMMOONN OOPPTTIIOONNSS

The options allowed when starting an rsync daemon are as follows:

----ddaaeemmoonn
This tells rsync that it is to run as a daemon. The daemon you start running may be accessed using an rsync client using the hhoosstt::::mmoodduullee or rrssyynncc::////hhoosstt//mmoodduullee// syntax.

If standard input is a socket then rsync will assume that it is being run via inetd, otherwise it will detach from the current terminal and become a background daemon. The daemon will read the config file (rsyncd.conf) on each connect made by a client and respond to requests accordingly. See the rrssyynnccdd..ccoonnff(5) man page for more details.

----aaddddrreessss
By default rsync will bind to the wildcard address when run as a daemon with the ----ddaaeemmoonn option. The ----aaddddrreessss option allows you to specify a specific IP address (or hostname) to bind to. This makes virtual hosting possible in conjunction with the ----ccoonnffiigg option. See also the "address" global option in the rsyncd.conf manpage.

----bbwwlliimmiitt==KKBBPPSS
This option allows you to specify a maximum transfer rate in kilobytes per second for the data the daemon sends. The client can still specify a smaller ----bbwwlliimmiitt value, but their requested value will be rounded down if they try to exceed it. See the client version of this option (above) for some extra details.

----ccoonnffiigg==FFIILLEE
This specifies an alternate config file than the default. This is only relevant when ----ddaaeemmoonn is specified. The default is /etc/rsyncd.conf unless the daemon is running over a remote shell program and the remote user is not the super-user; in that case the default is rsyncd.conf in the current directory (typically $HOME).

----nnoo--ddeettaacchh
When running as a daemon, this option instructs rsync to not detach itself and become a background process. This option is required when running as a service on Cygwin, and may also be useful when rsync is supervised by a program such as ddaaeemmoonnttoooollss or AIX's SSyyssttemm RReessoouurrccee CCoonnttrroolllleerr. ----nnoo--ddeettaacchh is also recommended when rsync is run under a debugger. This option has no effect if rsync is run from inetd or sshd.

----ppoorrtt==PPOORRTT
This specifies an alternate TCP port number for the daemon to listen on rather than the default of 873. See also the "port" global option in the rsyncd.conf manpage.

----lloogg--ffiillee==FFIILLEE
This option tells the rsync daemon to use the given log-file name instead of using the "log file" setting in the config file.

----lloogg--ffiillee--ffoorrmmaatt==FFOORRMMAATT
This option tells the rsync daemon to use the given FORMAT string instead of using the "log format" setting in the config

file. It also enables "transfer logging" unless the string is
empty, in which case transfer logging is turned off.

----ssoocckkooppttss
This overrides the ssoocckkeett oopptt iioonnss setting in the rsyncd.conf
file and has the same syntax.

--vv,, ----vveerrbboossee
This option increases the amount of information the daemon logs
during its startup phase. After the client connects, the dae-
mon's verbosity level will be controlled by the options that the
client used and the "max verbosity" setting in the module's con-
fig section.

--44,, ----iippvv44 or --66,, ----iippvv66
Tells rsync to prefer IPv4/IPv6 when creating the incoming sock-
ets that the rsync daemon will use to listen for connections.
One of these options may be required in older versions of Linux
to work around an IPv6 bug in the kernel (if you see an "address
already in use" error when nothing else is using the port, try
specifying ----iippvv66 or ----iippvv44 when starting the daemon).

--hh,, ----hheellpp
When specified after ----ddaaeemmoonn, print a short help page describ-
ing the options available for starting an rsync daemon.

FFIILLTTEERR RRUULLEESS
The filter rules allow for flexible selection of which files to trans-
fer (include) and which files to skip (exclude). The rules either

directly specify include/exclude patterns or they specify a way to
acquire more include/exclude patterns (e.g. to read them from a file).

As the list of files/directories to transfer is built, rsync checks
each name to be transferred against the list of include/exclude pat-
terns in turn, and the first matching pattern is acted on: if it is an
exclude pattern, then that file is skipped; if it is an include pattern
then that filename is not skipped; if no matching pattern is found,
then the filename is not skipped.

Rsync builds an ordered list of filter rules as specified on the com-
mand-line. Filter rules have the following syntax:

```
RULE [PATTERN_OR_FILENAME]
RULE,MODIFIERS [PATTERN_OR_FILENAME]
```

You have your choice of using either short or long RULE names, as
described below. If you use a short-named rule, the ',' separating the
RULE from the MODIFIERS is optional. The PATTERN or FILENAME that fol-
lows (when present) must come after either a single space or an under-
score (_). Here are the available rule prefixes:

```
eexxcclluuddee,, -- specifies an exclude pattern.
iinncclluuddee,, ++ specifies an include pattern.
mmeerrggee,, .. specifies a merge-file to read for more rules.
ddiirr--mmeerrggee,, :: specifies a per-directory merge-file.
hhiiddee,, HH specifies a pattern for hiding files from the transfer.
sshhooww,, SS files that match the pattern are not hidden.
pprrootteecctt,, PP specifies a pattern for protecting files from dele-
tion.
rriisskk,, RR files that match the pattern are not protected.
cclleeaarr,, !! clears the current include/exclude list (takes no arg)
```

When rules are being read from a file, empty lines are ignored, as are
comment lines that start with a "#".

Note that the ----iinncclluuddee/----eexxcclluuddee command-line options do not allow the
full range of rule parsing as described above -- they only allow the
specification of include/exclude patterns plus a "!" token to clear the
list (and the normal comment parsing when rules are read from a file).
If a pattern does not begin with "- " (dash, space) or "+ " (plus,
space), then the rule will be interpreted as if "+ " (for an include
option) or "- " (for an exclude option) were prefixed to the string. A
----ffiillltteerr option, on the other hand, must always contain either a short
or long rule name at the start of the rule.

Note also that the ----ffiillltteerr, ----iinncclluuddee, and ----eexxcclluuddee options take one
rule/pattern each. To add multiple ones, you can repeat the options on
the command-line, use the merge-file syntax of the ----ffiillltteerr option, or
the ----iinncclluuddee--ffrroomm/----eexxcclluuddee--ffrroomm options.

IINNCCLLUUDDEE//EEXXCCLLUUDDEE PPAATTTTEERRNN RRUULLEESS

You can include and exclude files by specifying patterns using the "+",
"-", etc. filter rules (as introduced in the FILTER RULES section
above). The include/exclude rules each specify a pattern that is
matched against the names of the files that are going to be trans-
ferred. These patterns can take several forms:

- if the pattern starts with a / then it is anchored to a particu-
 lar spot in the hierarchy of files, otherwise it is matched
 against the end of the pathname. This is similar to a

leading ^
in regular expressions. Thus "/foo" would match a file named
"foo" at either the "root of the transfer" (for a global rule)
or in the merge-file's directory (for a per-directory rule). An
unqualified "foo" would match any file or directory named "foo"
anywhere in the tree because the algorithm is applied recur-
sively from the top down; it behaves as if each path component
gets a turn at being the end of the file name. Even the unan-
chored "sub/foo" would match at any point in the hierarchy where
a "foo" was found within a directory named "sub". See the sec-
tion on ANCHORING INCLUDE/EXCLUDE PATTERNS for a full discussion
of how to specify a pattern that matches at the root of the
transfer.

- o if the pattern ends with a / then it will only match a direc-
tory, not a file, link, or device.

- o rsync chooses between doing a simple string match and wildcard
matching by checking if the pattern contains one of these three
wildcard characters: '*', '?', and '[' .

- o a '*' matches any non-empty path component (it stops at
slashes).

- o use '**' to match anything, including slashes.

- o a '?' matches any character except a slash (/).

- o a '[' introduces a character class, such as [a-z] or
[[:alpha:]].

- o in a wildcard pattern, a backslash can be used to escape a wild-
card character, but it is matched literally when no

wildcards
are present.

- if the pattern contains a / (not counting a trailing /) or a "**", then it is matched against the full pathname, including any leading directories. If the pattern doesn't contain a / or a "**", then it is matched only against the final component of the filename. (Remember that the algorithm is applied recursively so "full filename" can actually be any portion of a path from the starting directory on down.)

- a trailing "dir_name/***" will match both the directory (as if "dir_name/" had been specified) and all the files in the directory (as if "dir_name/**" had been specified). (This behavior is new for version 2.6.7.)

Note that, when using the ----rreeccuurrssiivvee (--rr) option (which is implied by --aa), every subcomponent of every path is visited from the top down, so include/exclude patterns get applied recursively to each subcomponent's full name (e.g. to include "/foo/bar/baz" the subcomponents "/foo" and "/foo/bar" must not be excluded). The exclude patterns actually short-circuit the directory traversal stage when rsync finds the files to send. If a pattern excludes a particular parent directory, it can render a deeper include pattern ineffectual because rsync did not descend through that excluded section of the hierarchy. This is particularly important when using a trailing '*' rule. For instance, this won't work:

```
+ /some/path/this-file-will-not-be-found
+ /file-is-included
```

```
- *
```

This fails because the parent directory "some" is excluded by the '*' rule, so rsync never visits any of the files in the "some" or "some/path" directories. One solution is to ask for all directories in the hierarchy to be included by using a single rule: "+ */" (put it somewhere before the "- *" rule), and perhaps use the ----pprruunnee--eemmppttyy--ddiirrss option. Another solution is to add specific include rules for all the parent dirs that need to be visited. For instance, this set of rules works fine:

```
+ /some/
+ /some/path/
+ /some/path/this-file-is-found
+ /file-also-included
- *
```

Here are some examples of exclude/include matching:

- "- *.o" would exclude all filenames matching *.o
- "- /foo" would exclude a file (or directory) named foo in the transfer-root directory
- "- foo/" would exclude any directory named foo
- "- /foo/*/bar" would exclude any file named bar which is at two levels below a directory named foo in the transfer-root direc-tory
- "- /foo/**/bar" would exclude any file named bar two or more levels below a directory named foo in the transfer-root direc-tory
- The combination of "+ */", "+ *.c", and "- *" would include

all
directories and C source files but nothing else (see also the
----pprruunnee--eemmppttyy--ddiirrss option)

o The combination of "+ foo/", "+ foo/bar.c", and "- *" would
include only the foo directory and foo/bar.c (the foo directory
must be explicitly included or it would be excluded by the "*")

MMEERRGGEE--FFIILLEE FFIILLTTEERR RRUULLEESS

You can merge whole files into your filter rules by specifying either a
merge (.) or a dir-merge (:) filter rule (as introduced in the FILTER
RULES section above).

There are two kinds of merged files -- single-instance ('.') and per-
directory (':'). A single-instance merge file is read one time, and
its rules are incorporated into the filter list in the place of the "."
rule. For per-directory merge files, rsync will scan every directory
that it traverses for the named file, merging its contents when the
file exists into the current list of inherited rules. These per-direc-
tory rule files must be created on the sending side because it is the
sending side that is being scanned for the available files to transfer.
These rule files may also need to be transferred to the receiving side
if you want them to affect what files don't get deleted (see PER-DIREC-
TORY RULES AND DELETE below).

Some examples:

```
merge /etc/rsync/default.rules
. /etc/rsync/default.rules
dir-merge .per-dir-filter
dir-merge,n- .non-inherited-per-dir-excludes
:n- .non-inherited-per-dir-excludes
```

The following modifiers are accepted after a merge or dir-merge rule:

- A -- specifies that the file should consist of only exclude patterns, with no other rule-parsing except for in-file comments.
- A ++ specifies that the file should consist of only include patterns, with no other rule-parsing except for in-file comments.
- A CC is a way to specify that the file should be read in a CVS-compatible manner. This turns on 'n', 'w', and '-', but also allows the list-clearing token (!) to be specified. If no filename is provided, ".cvsignore" is assumed.
- A ee will exclude the merge-file name from the transfer; e.g. "dir-merge,e .rules" is like "dir-merge .rules" and "- .rules".
- An nn specifies that the rules are not inherited by subdirectories.
- A ww specifies that the rules are word-split on whitespace instead of the normal line-splitting. This also turns off comments. Note: the space that separates the prefix from the rule is treated specially, so "- foo + bar" is parsed as two rules (assuming that prefix-parsing wasn't also disabled).
- You may also specify any of the modifiers for the "+" or "-" rules (below) in order to have the rules that are read in from the file default to having that modifier set. For instance, "merge,-/ .excl" would treat the contents of .excl as

absolute-
path excludes, while "dir-merge,s .filt" and ":sC" would each
make all their per-directory rules apply only on the sending
side.

The following modifiers are accepted after a "+" or "-":

- A "/" specifies that the include/exclude rule should be matched against the absolute pathname of the current item. For example, "-/ /etc/passwd" would exclude the passwd file any time the transfer was sending files from the "/etc" directory, and "-/ subdir/foo" would always exclude "foo" when it is in a dir named "subdir", even if "foo" is at the root of the current transfer.
- A "!" specifies that the include/exclude should take effect if the pattern fails to match. For instance, "-! */" would exclude all non-directories.
- A CC is used to indicate that all the global CVS-exclude rules should be inserted as excludes in place of the "-C". No arg should follow.
- An ss is used to indicate that the rule applies to the sending side. When a rule affects the sending side, it prevents files from being transferred. The default is for a rule to affect both sides unless ----ddeelleettee--eexxcclluuddeedd was specified, in which case default rules become sender-side only. See also the hide (H) and show (S) rules, which are an alternate way to specify send-ing-side includes/excludes.

o An rr is used to indicate that the rule applies to the receiving side. When a rule affects the receiving side, it prevents files from being deleted. See the ss modifier for more info. See also the protect (P) and risk (R) rules, which are an alternate way to specify receiver-side includes/excludes.

Per-directory rules are inherited in all subdirectories of the direc-
tory where the merge-file was found unless the 'n' modifier was used.
Each subdirectory's rules are prefixed to the inherited per-directory
rules from its parents, which gives the newest rules a higher priority
than the inherited rules. The entire set of dir-merge rules are
grouped together in the spot where the merge-file was specified, so it
is possible to override dir-merge rules via a rule that got specified
earlier in the list of global rules. When the list-clearing rule ("!")
is read from a per-directory file, it only clears the inherited rules
for the current merge file.

Another way to prevent a single rule from a dir-merge file from being
inherited is to anchor it with a leading slash. Anchored rules in a
per-directory merge-file are relative to the merge-file's directory, so
a pattern "/foo" would only match the file "foo" in the directory where
the dir-merge filter file was found.

Here's an example filter file which you'd specify via ---- ffiilltteerr==""..
ffiillee"":

```
merge /home/user/.global-filter
- *.gz
dir-merge .rules
+ *.[ch]
- *.o
```

This will merge the contents of the /home/user/.global-filter file at the start of the list and also turns the ".rules" filename into a per-directory filter file. All rules read in prior to the start of the directory scan follow the global anchoring rules (i.e. a leading slash matches at the root of the transfer).

If a per-directory merge-file is specified with a path that is a parent directory of the first transfer directory, rsync will scan all the parent dirs from that starting point to the transfer directory for the indicated per-directory file. For instance, here is a common filter (see --FF):

```
--filter=': /.rsync-filter'
```

That rule tells rsync to scan for the file .rsync-filter in all directories from the root down through the parent directory of the transfer prior to the start of the normal directory scan of the file in the directories that are sent as a part of the transfer. (Note: for an rsync daemon, the root is always the same as the module's "path".)

Some examples of this pre-scanning for per-directory files:

```
rsync -avF /src/path/ /dest/dir
rsync -av --filter=': ../../.rsync-filter' /src/path/ /dest/dir
rsync -av --filter=': .rsync-filter' /src/path/ /dest/dir
```

The first two commands above will look for ".rsync-filter" in "/" and "/src" before the normal scan begins looking for the file in "/src/path" and its subdirectories. The last command avoids the parent-dir scan and only looks for the ".rsync-filter" files in

each
directory that is a part of the transfer.

If you want to include the contents of a ".cvsignore" in your patterns,
you should use the rule ":C", which creates a dir-merge of the .cvsig-
nore file, but parsed in a CVS-compatible manner. You can use this to
affect where the ----ccvvss--eexxcclluuddee (--CC) option's inclusion of the per-
directory .cvsignore file gets placed into your rules by putting the
":C" wherever you like in your filter rules. Without this, rsync would
add the dir-merge rule for the .cvsignore file at the end of all your
other rules (giving it a lower priority than your command-line rules).
For example:

```
cat <<EOT | rsync -avC --filter=. -' a/ b
+ foo.o
:C
- *.old
EOT
rsync -avC --include=foo.o -f :C --exclude='*.old' a/ b
```

Both of the above rsync commands are identical. Each one will merge
all the per-directory .cvsignore rules in the middle of the list rather
than at the end. This allows their dir-specific rules to supersede the
rules that follow the :C instead of being subservient to all your
rules. To affect the other CVS exclude rules (i.e. the default list of
exclusions, the contents of $HOME/.cvsignore, and the value of $CVSIG-
NORE) you should omit the --CC command-line option and instead insert a
"-C" rule into your filter rules; e.g. "--filter=-C".

LLIISSTT--CCLLEEAARRIINNGG FFIILLTTEERR RRUULLEE

You can clear the current include/exclude list by using the "!" filter
rule (as introduced in the FILTER RULES section above). The

"current"
list is either the global list of rules (if the rule is encountered
while parsing the filter options) or a set of per-directory rules
(which are inherited in their own sub-list, so a subdirectory can use
this to clear out the parent's rules).

AANNCCHHOORRIINNGG IINNCCLLUUDDEE//EEXXCCLLUUDDEE PPAATTTTEERRNNSS
As mentioned earlier, global include/exclude patterns are anchored at
the "root of the transfer" (as opposed to per-directory patterns, which
are anchored at the merge-file's directory). If you think of the
transfer as a subtree of names that are being sent from sender to
receiver, the transfer-root is where the tree starts to be duplicated
in the destination directory. This root governs where patterns that
start with a / match.

Because the matching is relative to the transfer-root, changing the
trailing slash on a source path or changing your use of the ----
rreellaattiivvee
option affects the path you need to use in your matching (in addition
to changing how much of the file tree is duplicated on the destination
host). The following examples demonstrate this.

Let's say that we want to match two source files, one with an absolute
path of "/home/me/foo/bar", and one with a path of "/home/you/bar/baz".
Here is how the various command choices differ for a 2-source transfer:

```
Example cmd: rsync -a /home/me /home/you /dest
+/- pattern: /me/foo/bar
+/- pattern: /you/bar/baz
Target file: /dest/me/foo/bar
Target file: /dest/you/bar/baz


Example cmd: rsync -a /home/me/ /home/you/ /dest
```

```
              +/- pattern: /foo/bar               (note missing "me")
              +/- pattern: /bar/baz               (note missing "you")
              Target file: /dest/foo/bar
              Target file: /dest/bar/baz


              Example cmd: rsync -a --relative /home/me/ /home/you /dest
              +/- pattern: /home/me/foo/bar       (note full path)
              +/- pattern: /home/you/bar/baz      (ditto)
              Target file: /dest/home/me/foo/bar
              Target file: /dest/home/you/bar/baz


              Example cmd: cd /home; rsync -a --relative me/foo you/ /dest
              +/- pattern: /me/foo/bar      (starts at specified path)
              +/- pattern: /you/bar/baz     (ditto)
              Target file: /dest/me/foo/bar
              Target file: /dest/you/bar/baz
```

The easiest way to see what name you should filter is to just look at the output when using ----vveerrbboossee and put a / in front of the name (use the ----ddrryy--rruunn option if you're not yet ready to copy any files).

PPEERR--DDIIRREECCTTOORRYY RRUULLEESS AANNDD DDEELLEETTEE

Without a delete option, per-directory rules are only relevant on the sending side, so you can feel free to exclude the merge files themselves without affecting the transfer. To make this easy, the 'e' modifier adds this exclude for you, as seen in these two equivalent commands:

```
rsync -av --filter=': .excl' --exclude=.excl host:src/dir /dest
rsync -av --filter=':e .excl' host:src/dir /dest
```

However, if you want to do a delete on the receiving side AND you want some files to be excluded from being deleted, you'll need to be sure that the receiving side knows what files to exclude. The easiest way

is to include the per-directory merge files in the transfer and use
----ddeelleettee--aafftteerr, because this ensures that the receiving side gets all
the same exclude rules as the sending side before it tries to delete
anything:

```
rsync -avF --delete-after host:src/dir /dest
```

However, if the merge files are not a part of the transfer, you'll need
to either specify some global exclude rules (i.e. specified on the com-
mand line), or you'll need to maintain your own per-directory merge
files on the receiving side. An example of the first is this (assume
that the remote .rules files exclude themselves):

```
rsync -av --filter=': .rules' --filter='. /my/extra.rules'
   --delete host:src/dir /dest
```

In the above example the extra.rules file can affect both sides of the
transfer, but (on the sending side) the rules are subservient to the
rules merged from the .rules files because they were specified after
the per-directory merge rule.

In one final example, the remote side is excluding the .rsync-filter
files from the transfer, but we want to use our own .rsync-filter files
to control what gets deleted on the receiving side. To do this we must
specifically exclude the per-directory merge files (so that they don't
get deleted) and then put rules into the local files to control what
else should not get deleted. Like one of these commands:

```
rsync -av --filter=':e /.rsync-filter' --delete \
    host:src/dir /dest
rsync -avFF --delete host:src/dir /dest
```

BBAATTCCHH MMOODDEE

Batch mode can be used to apply the same set of updates to many identi-
cal systems. Suppose one has a tree which is replicated on a number of
hosts. Now suppose some changes have been made to this source tree and
those changes need to be propagated to the other hosts. In order to do
this using batch mode, rsync is run with the write-batch option to
apply the changes made to the source tree to one of the destination
trees. The write-batch option causes the rsync client to store in a
"batch file" all the information needed to repeat this operation
against other, identical destination trees.

To apply the recorded changes to another destination tree, run rsync
with the read-batch option, specifying the name of the same batch file,
and the destination tree. Rsync updates the destination tree using the
information stored in the batch file.

For convenience, one additional file is creating when the write-batch
option is used. This file's name is created by appending ".sh" to the
batch filename. The .sh file contains a command-line suitable for
updating a destination tree using that batch file. It can be executed
using a Bourne (or Bourne-like) shell, optionally passing in an alter-
nate destination tree pathname which is then used instead of the origi-
nal path. This is useful when the destination tree path differs from
the original destination tree path.

Generating the batch file once saves having to perform the file status,
checksum, and data block generation more than once when updating multi-
ple destination trees. Multicast transport protocols can be used to

transfer the batch update files in parallel to many hosts at once, instead of sending the same data to every host individually.

Examples:

```
$ rsync --write-batch=foo -a host:/source/dir/ /adest/dir/
$ scp foo* remote:
$ ssh remote ./foo.sh /bdest/dir/
```

```
$ rsync --write-batch=foo -a /source/dir/ /adest/dir/
$ ssh remote rsync --read-batch=- -a /bdest/dir/ <foo
```

In these examples, rsync is used to update /adest/dir/ from /source/dir/ and the information to repeat this operation is stored in "foo" and "foo.sh". The host "remote" is then updated with the batched data going into the directory /bdest/dir. The differences between the two examples reveals some of the flexibility you have in how you deal with batches:

- The first example shows that the initial copy doesn't have to be local -- you can push or pull data to/from a remote host using either the remote-shell syntax or rsync daemon syntax, as desired.
- The first example uses the created "foo.sh" file to get the right rsync options when running the read-batch command on the remote host.
- The second example reads the batch data via standard input so that the batch file doesn't need to be copied to the remote machine first. This example avoids the foo.sh script because it needed to use a modified ----rreeaadd--bbaattcchh option, but you could edit

the script file if you wished to make use of it (just be sure
that no other option is trying to use standard input, such as
the "----eexxcclluuddee--ffrroomm==--" option).

Caveats:

The read-batch option expects the destination tree that it is updating
to be identical to the destination tree that was used to create the
batch update fileset. When a difference between the destination trees
is encountered the update might be discarded with a warning (if the
file appears to be up-to-date already) or the file-update may be
attempted and then, if the file fails to verify, the update discarded
with an error. This means that it should be safe to re-run a read-
batch operation if the command got interrupted. If you wish to force
the batched-update to always be attempted regardless of the file's size
and date, use the --II option (when reading the batch). If an error
occurs, the destination tree will probably be in a partially updated
state. In that case, rsync can be used in its regular (non-batch) mode
of operation to fix up the destination tree.

The rsync version used on all destinations must be at least as new as
the one used to generate the batch file. Rsync will die with an error
if the protocol version in the batch file is too new for the batch-
reading rsync to handle. See also the ----pprrootocool option for a way to
have the creating rsync generate a batch file that an older rsync can
understand. (Note that batch files changed format in version 2.6.3, so
mixing versions older than that with newer versions will not work.)

When reading a batch file, rsync will force the value of

certain
options to match the data in the batch file if you didn't set them to
the same as the batch-writing command. Other options can (and should)
be changed. For instance ----wwrriittee--bbaattcchh changes to ----rreeaadd--bbaattcchh,
----ffiilleess--ffrroomm is dropped, and the ----ffiilltteerr/----iinncclluuddee/----eexxcclluuddee options
are not needed unless one of the ----ddeelleettee options is specified.

The code that creates the BATCH.sh file transforms any fil-
ter/include/exclude options into a single list that is appended as a
"here" document to the shell script file. An advanced user can use
this to modify the exclude list if a change in what gets deleted by
----ddeelleettee is desired. A normal user can ignore this detail and just use
the shell script as an easy way to run the appropriate ----rreeaadd--bbaattcchh
command for the batched data.

The original batch mode in rsync was based on "rsync+", but the latest
version uses a new implementation.

SSYYMMBBOOLLIICC LLIINNKKSS

Three basic behaviors are possible when rsync encounters a symbolic
link in the source directory.

By default, symbolic links are not transferred at all. A message
"skipping non-regular" file is emitted for any symlinks that exist.

If ----lliinnkkss is specified, then symlinks are recreated with the same tar-
get on the destination. Note that ----aarrcchhiivvee implies ----lliinnkkss.

If ----ccooppyy--lliinnkkss is specified, then symlinks are "collapsed" by copying
their referent, rather than the symlink.

rsync also distinguishes "safe" and "unsafe" symbolic links. An

exam-
 ple where this might be used is a web site mirror that wishes
ensure
 the rsync module they copy does not include symbolic links
to
 //eettcc//ppaasssswwdd in the public section of the
site. Using
 ----ccooppyy--uunnssaaffee--lliinnkkss will cause any links to be
copied as the file they
 point to on the destination. Using ----ssaaffee--lliinnkkss will
cause unsafe
 links to be omitted altogether. (Note that you must specify ----
lliinnkkss
 for ----ssaaffee--lliinnkkss to have any effect.)

 Symbolic links are considered unsafe if they are absolute
symlinks
 (start with //), empty, or if they contain enough ""....""
components to
 ascend from the directory being copied.

 Here's a summary of how the symlink options are interpreted. The
list
 is in order of precedence, so if your combination of options isn't
men-
 tioned, use the first line that is a complete subset of your
options:

 ----ccooppyy--lliinnkkss
 Turn all symlinks into normal files (leaving no symlinks for
any
 other options to affect).

 ----lliinnkkss ----ccooppyy--uunnssaaffee--lliinnkkss
 Turn all unsafe symlinks into files and duplicate all safe
sym-
 links.

 ----ccooppyy--uunnssaaffee--lliinnkkss
 Turn all unsafe symlinks into files, noisily skip all safe
sym-
 links.

 ----lliinnkkss ----ssaaffee--lliinnkkss
 Duplicate safe symlinks and skip unsafe ones.

----lliinnkkss
Duplicate all symlinks.

DDIIAAGGNNOOSSTTIICCSS

rsync occasionally produces error messages that may seem a little cryp-
tic. The one that seems to cause the most confusion is "protocol ver-
sion mismatch -- is your shell clean?".

This message is usually caused by your startup scripts or remote shell
facility producing unwanted garbage on the stream that rsync is using
for its transport. The way to diagnose this problem is to run your
remote shell like this:

ssh remotehost /bin/true > out.dat

then look at out.dat. If everything is working correctly then out.dat
should be a zero length file. If you are getting the above error from
rsync then you will probably find that out.dat contains some text or
data. Look at the contents and try to work out what is producing it.
The most common cause is incorrectly configured shell startup scripts
(such as .cshrc or .profile) that contain output statements for non-
interactive logins.

If you are having trouble debugging filter patterns, then try specify-
ing the --vvvv option. At this level of verbosity rsync will show why
each individual file is included or excluded.

EEXXIITT VVAALLUUEESS

00 Success

11 Syntax or usage error

22 Protocol incompatibility

33 Errors selecting input/output files, dirs

44 Requested action not supported: an attempt was made to manipu-
late 64-bit files on a platform that cannot support them; or an
option was specified that is supported by the client and not by
the server.

55 Error starting client-server protocol

66 Daemon unable to append to log-file

1100 Error in socket I/O

1111 Error in file I/O

1122 Error in rsync protocol data stream

1133 Errors with program diagnostics

1144 Error in IPC code

2200 Received SIGUSR1 or SIGINT

2211 Some error returned by waitpid()

2222 Error allocating core memory buffers

2233 Partial transfer due to error

2244 Partial transfer due to vanished source files

2255 The --max-delete limit stopped deletions

3300 Timeout in data send/receive

EENNVVIIRROONNMMEENNTT VVAARRIIAABBLLEESS

CCVVSSIIGGNNOORREE
The CVSIGNORE environment variable supplements any ignore pat-
terns in .cvsignore files. See the ----ccvvss--eexxcclluuddee option for more
details.

RRSSYYNNCC__RRSSHH

The RSYNC_RSH environment variable allows you to override the default shell used as the transport for rsync. Command line options are permitted after the command name, just as in the --ee option.

RRSSYYNNCC__PPRROOXXYY
The RSYNC_PROXY environment variable allows you to redirect your rsync client to use a web proxy when connecting to a rsync dae-mon. You should set RSYNC_PROXY to a hostname:port pair.

RRSSYYNNCC__PPAASSSSWWOORRDD
Setting RSYNC_PASSWORD to the required password allows you to run authenticated rsync connections to an rsync daemon without user intervention. Note that this does not supply a password to a shell transport such as ssh.

UUSSEERR or LLOOGGNNAAMMEE
The USER or LOGNAME environment variables are used to determine the default username sent to an rsync daemon. If neither is set, the username defaults to "nobody".

HHOOMMEE The HOME environment variable is used to find the user's default .cvsignore file.

FFIILLEESS

/etc/rsyncd.conf or rsyncd.conf

SSEEEE AALLSSOO

rrssyynnccdd..ccoonnff(5) ffccnnttll(2)

BBUUGGSS

times are transferred as *nix time_t values

When transferring to FAT filesystems rsync may re-sync unmodified

files. See the comments on the ----mmooddiiffyy--wwiinnddooww option.

file permissions, devices, etc. are transferred as native numerical values

see also the comments on the ----ddeelleettee option

Please report bugs! See the website at http://rsync.samba.org/

VVEERRSSIIOONN

This man page is current for version 2.6.9 of rsync.

IINNTTEERRNNAALL OOPPTTIIOONNSS

The options ----sseerrvveerr and ----sseennddeerr are used internally by rsync, and should never be typed by a user under normal circumstances. Some awareness of these options may be needed in certain scenarios, such as when setting up a login that can only run an rsync command. For instance, the support directory of the rsync distribution has an example script named rrsync (for restricted rsync) that can be used with a restricted ssh login.

CCRREEDDIITTSS

rsync is distributed under the GNU public license. See the file COPYING for details.

A WEB site is available at http://rsync.samba.org/. The site includes an FAQ-O-Matic which may cover questions unanswered by this manual page.

The primary ftp site for rsync is ftp://rsync.samba.org/pub/rsync.

We would be delighted to hear from you if you like this program.

This program uses the excellent zlib compression library written by Jean-loup Gailly and Mark Adler.

TTHHAANNKKSS

Thanks to Richard Brent, Brendan Mackay, Bill Waite, Stephen Rothwell and David Bell for helpful suggestions, patches and testing of rsync.
I've probably missed some people, my apologies if I have.

Especial thanks also to: David Dykstra, Jos Backus, Sebastian Krahmer, Martin Pool, Wayne Davison, J.W. Schultz.

AAUUTTHHOORR

rsync was originally written by Andrew Tridgell and Paul Mackerras.
Many people have later contributed to it.

Mailing lists for support and development are available at http://lists.samba.org